

开源时代

Open SourceTime

2010年五月刊 总第二十期

开源业界

MeeGo步入互联网战局

深入分析: Flash VS HTML5 谁统江山

技术新知

Java_7新IO特性解析

邮件服务Postfix详解

社区扫描

Ubuntu10.04 LTS新功能阐释

尘埃落定 法院判定Novell拥有Unix版权

行业观察

从开源到收费: 甲骨文榨取Solaris最后价值

Java之父黯然离职 开源将何去何从

本期推荐:

谁是MySQL的主人

——2010全球用户大会观感

谁是MySQL的新主人? 这似乎是个伪命题, MySQL的新主人无疑是那个强势的甲骨文公司。去年对Sun的收购, 让甲骨文顺利的将一个潜在的数据库对手收入囊中, 开源社区大哗, 一时众说纷纭, 唱衰者有之、看好者也有之, 但总体看来, 忧虑的情绪在社区弥散。数位MySQL创始人的“拯救”行动, 更将MySQL的悲壮推向高潮。

内容目录

刊首语.....	4
开源业界.....	5
红帽 Linux 发行版项目组介绍 RHEL6 beta 版	5
Google 确认 Android 2.2 内置对 Flash 的支持	7
挑战微软 Office OpenOffice.org 发布 3.2 正式版	8
Ubuntu 操作系统商业发展策略剖析	9
细数 Office 2010 十大值得关注的新特性.....	11
红旗软件成功通过 IBM WAS 6.0/7.0 认证.....	13
MySQL 陷内忧外患已处于消亡的边缘?	14
深入分析: Flash VS HTML5 到底谁统江山	16
Novell 和 Red Hat 在用户界面专利案中获胜	20
开源漫谈之 GNU GPL 的前世今生	20
资深 Linux 系统使用者的玩具清单	24
MeeGo 步入互联网战局	26
FLOW 让你提前尝试 Google 的云操作系统	28
IBM 收购云计算软件商 Cast Iron	28
Eclipse 宣布新的 SOA 平台启动	28
赛门铁克: Linux 垃圾邮件威胁严重	29
社区扫描.....	31
Ubuntu 10.04 LTS 新功能阐释	31
最封闭的开源系统: 话说 Android 的八宗罪	32
尘埃落定 法院判定 Novell 拥有 Unix 版权	34
Facebook 推出在线文档共享网站 Docs.com	35
Oracle 收购 Sun 并不总是坏事的 8 大原因	35
Linux 与 Mac OS 赛跑, 谁领先?	37
Google 在扼杀开源贡献者?	38
白宫向开源项目捐赠代码	38
行业观察.....	39
MySQL 数据库研发团队女掌门加盟 EnterpriseDB	39
RIA 之战 微软欲借开源策略后来居上?	39
iPhone SDK4 惹怒开发者 老乔态度强硬	43
Facebook 开放图谱 API 引争议: 被指非真正开放	44
Linux 基金会董事: 未来 2-5 年硬件将会免费	44
Java 之父黯然离职 开源将何去何从	46
从开源到收费: 甲骨文榨取 Solaris 最后价值.....	46
大型机 Linux: 风雨十周年	47
本期推荐.....	49
谁是 MySQL 的新主人.....	49

技术新知.....	52
Linux 驱动开发方法论.....	52
Java 7 新 I/O 特性解析.....	61
邮件服务 postfix 详解.....	66
ARM-Linux 使用 SD 卡根文件系统	68
OpenSSL 库中 des 加密函数库的使用.....	72
Linux 系统管理员应该知道的 20 个系统监控工具	74
网友热评.....	87

本期编辑：林惠菊 周荣茂 覃里

美工：林惠菊

编校：林惠菊

投稿邮箱：rmzhou@staff.chinaunix.net

本刊网址：<http://linux.chinaunix.net/ebook/>

刊首语

《开源时代》2010 年第五期迟到了这么久，真是非常抱歉！因为编辑团队的调整，导致这期内容只到现在才发布。但愿迟到的杂志能够给大家带来一丝慰藉。

本月最值得关注的可能就是 Ubuntu 10.04 的发布，在 10 月 31 号最终发布的 Ubuntu 10.04 没有让大家失望，带来了大量的新功能和特性。Ubuntu 10.04 LTS 的最大卖点是启动速度有了极大的提升。据官方介绍是可以在上网本上实现 10 秒内完全启动。虽然距最后发布期限一天内发现一个重大 bug，但是依赖于社区的强大的 bug 修复机制，很快在 12 小时候内解决问题并重新打包和发行新的 ISO 下载镜像。更多相关内容可以关注本期社区扫描栏目中相关内容。

Android 风头正劲的时候，批评声也越来越多。Google 采用了一系列的控制手段来保证每一部 Android 手机上都有它指定的软件和硬件规格。然而，他们同时又利用 Android SDK 里面的 Apache 许可证来大肆鼓吹 Android 是开放的。

虽然饱受质疑，但是 Google 的 Android 移动平台已经在移动行业得到了运营商和手机厂商的广泛支持，仅剩固执的诺基亚。于是联合 Intel 正式对 Android 宣传，对外宣布 Meego 项目。在 4 月份举行的 Intel IDF 北京上，MeeGo 隆重推出相关技术路线图，并发布了高端合作计划。

Oracle 收购 Sun 的余波还没有结束，大规模的裁员也拉开了大幕，亚洲和欧洲是重灾区，其中 MySQL 和 Sun 的开源部门首当其冲，作为 Sun 的优质开源资产，MySQL 的前途令人担忧。之前的 MySQL 研发团队负责人直接跳槽到竞争对手—EnterpriseDB 公司。也许以后的 MySQL 将会有两个流派，一方面是 Oracle 公司支持的庙堂之上，一个是社区推崇的江湖之远。无论如何，两者都没有把 MySQL 变坏的打算。

值得欣慰的是，又有新的同事加入到《开源时代》杂志的团队中来，希望杂志能够重新步入正轨，也希望大家能够给我们更多更好的建议！

----ChinaUnix 社区编辑：周荣茂

开源业界

红帽 Linux 发行版项目组介绍 RHEL6 beta 版

我们怀着兴奋的心情告诉您一个好消息，红帽企业 Linux 6 Beta 版今日向公众推出，这是我们发布下一个重要红帽企业 Linux 平台的第一步。从今天开始，我们诚邀我们的客户、合作伙伴和公众成员安装测试迄今为止我们最雄心勃勃、最重要的操作平台的操作系统，并为我们提供反馈。

红帽企业 Linux 第一版已经发布近 8 年了。自发布之日起，该产品确立了作为领先企业级开源操作系统之一的地位。安装的系统应用在从便携机到大型机各种机器中，帮助确立了质量、认证的基础设施、长期的稳定性、性能和安全性的标杆。从主体街到华尔街，红帽企业 Linux 几乎进入到各行各业中。



在红帽企业 Linux 进入 Beta 测试之时，当前支持的版本：红帽企业 Linux 5，依然是红帽软件产品资产组合的基石。红帽企业 Linux 5 是在 2007 年 3 月首次发布的，自发布后，经历了多次定期更新。上个月，我们刚刚发布了红帽企业 Linux 5 的第 5 次更新，增加了新的特性和硬件支持。在 2014 年之前，红帽企业 Linux 5 平台将继续得到红帽公司以及其 ISV 和 OEM 合作伙伴的支持。

展望未来，红帽企业 Linux 6 模糊了虚拟、物理和云计算之间的界线，以适应当代 IT 环境中发生的转变。红帽企业 Linux 6 从内核到应用基础设施到开发工具链都采用了升级的核心技术，可以满足未来几代硬件、软件技术的需要。

新版本的主旋律包括无处不在的虚拟化、更好的稳定性和高可用性、更高的能效以及提供多个最新软件技术。在今日 Beta 版发布之际，我们将介绍几项新的值得注意的改进：

全面的电源管理能力

按时的内核改进使系统可以更频繁地将没有活动任务的处理器变为空闲状态。这将导致比以前版本温度更低的 CPU 和更高的节电。Powertop 等新监测工具可以帮助确定可以解决的能耗问题，从而进一步减少能耗。像 “tuned” 这样的新调节工具（一种自适应系统调节后台程序）使系统可以根据服务使用模式的分析调节能耗。



性能改进

红帽工程师在我们计划出现在红帽企业 Linux 6 中的各种内核性能改进的上游开发中发挥着关键作用，完全重写进程调度程序，使它可以通过让更高优先级的进程在最低限度的较低优先级处理干扰的条件下，更公平地在处理器之间分配计算时间。此外，还进行了多种多处理器锁同步改进。例如，消除不必要的锁定事件、用睡眠锁定代替许多旋转（spin）锁定和采用更高效的锁定基元。这些根本的变化影响到许多内核子系统。

可伸缩性改进

新推出的硬件导致了商品计算平台的重大发展。例如，现在一台 5U 机架式机柜中可容纳 64 个 CPU 和 2TB 内存。这些系统以及它们的后继产品将要达到红帽企业 Linux 5 的可伸缩性极限。红帽企业 Linux 6 的一个主要特性是：它可以提供适应未来系统的可伸缩性。其可伸缩性能力从对大量 CPU 和内存配置的优化的支持到处理更多数量的系统互联总线 and 外设的能力。在虚拟化变得同裸机部署一样无处不在之时，这些能力适合于裸机环境和虚拟化环境。

新安全特性

一种叫做系统安全服务后台程序（SSSD）的新服务提供对身份的集中管理。它还具有缓存证书供离线之用的能力。新 SE Linux 沙箱特性使得不可信的内容可以在一个不会影响到系统其余部分的隔离的环境中执行。这包括隔离任何运行在红帽企业 Linux 6 上的虚拟客户机的能力。

资源管理

在一种叫做控制组（即 cgroups）的新框架的帮助下，新系统提供对硬件资源的细颗粒度控制、

分配和管理。cgroups 运行在进程组水平上，可被用于为应用管理从 CPU、内存、网络 and 硬盘 I/O 的资源。该框架还被用于管理虚拟客户机。

虚拟化

红帽企业 Linux 6 扩展了较早的红帽企业 Linux 版本提供的集成的基于 KVM 的虚拟化技术。新系统具有多个性能、调度程序和硬件支持的改进，提供无论采用什么部署模型的更好的灵活性和控制。

存储

通过 FCoE 和 iSCSI 协议对网络块存储的支持，使利用 LVM/DM 执行在线改变镜像的和多路径的卷大小成为可能。

文件系统

新版系统包括 ext4 文件系统。作为下一代扩展文件系统族，它包括对更大文件尺寸的支持、效率更高的硬盘空间分配、更好的文件系统检查和更强健的日志。除了 ext4 外，我们还打算提供 XFS 文件系统。XFS 适用于超大的文件和目录，包括像清除碎片和在文件系统使用时改变文件系统大小的能力。NFS 已经级升到了版本 4，从而包括对 IPv6 的支持。

可靠性、可用性和适用性 (RAS)

新版本利用新硬件能力来提供像热添加设备和硬件以及通过 AER 的 PCIe 设备的增强错误检查等特性。它还将包括高级数据完整性特性 (DIF/DIX)。这类特性通过硬件检查和检验来自应用的数据。ABRT (自动缺陷报告工具) 的引进提供了确定和报告系统异常情况——如内核故障 (kernel oops) 和用户空间应用崩溃——的更一致的方式。

编译器和工具

GCC 编译器已经升级到版本 4.4。这一版本遵照 C++ 0x 草案标准进行编译。它还符合 OpenMP 3.0，包括许多调试功能。SystemTap 改进包括对用户空间探测的更好的支持、更安全的脚本编译服务器和使非根用户可以访问 SystemTap 的新的非特权模式。此外，新编译器还有许多其它已经升级到最新版本的库和更多的语言和运行环境，包括完整的 LAMP 栈和 OpenJDK。

桌面

新版本引进了对显示类型的检测和对多种显示器的支持。我们还增加了支持 NVIDIA 图形设备的升级的新驱动程序。当然，如果不对 GNOME 和 KDE 桌面进行更新，新版本将是不完全的。

红帽企业 Linux 6 中的各种技术将随着对关键硬件平台的扩展支持提供，我们认为这些技术将使新版平台成为吸引新老客户的诱人的选择。同以往一样，新版本的一部分价值在于我们的企业认证。

目前，成千上万种应用通过了在红帽企业 Linux 上运行的认证，不管它运行在“裸机”部署、虚拟化部署，还是云部署中。这就使红帽企业 Linux 成为客户和合伙的首选操作系统。

Google 确认 Android 2.2 内置对 Flash 的支持

继 Chrome 内置 Flash 后，Google 再次宣布，将在 Android 2.2 (Froyo) 系统中添加对 Adobe Flash 10.1 的支持，这套系统将在数周内的 Google I/O 上发布。



之前，苹果不断以性能不行，占内存过高和“封闭标准”为由将 Flash 拒之门外，而 Google 则对 Flash 表现出极大的包容态度，Google 的工程师 Andy Rubin 表示“开放总能带来好处”--同样是一个“开放”，Google 和苹果价值观差异可见一斑。

挑战微软 Office OpenOffice.org 发布 3.2 正式版

诚然微软的 Office 在办公市场有很大的份额，但是来自开源社区的 OpenOffice.org 一直也在努力，从刚开始对中文支持都不完善，到现在的和 MS Office 不相上下，OpenOffice.org 用实际行动证明了开源社区的力量，而且也用它的跨平台特性获得了不少用户的亲睐。



前一阵微软旗下著名办公软件 Office 发布了最新 2010 版本，提供了大量新特性和更新，而本次 OpenOffice.org 的更新似乎是要和微软进行正面抗衡，那么让我们看看本次版本更新带来了什么特性吧。

首先 3.2 版本的 OpenOffice.org 整体提升有以下几点：

- 大幅度的提升了启动速度，官方表示相比 3.0 版本，节约了 46% 的启动时间。
- 提升了对 ODF (Open Document Format) 文件的支持。
- 增加了对新版本文件格式的支持，例如最新的 MS Office XML 格式文件 (*.docx, *.dotx, *.dotm, *.xlsx, *.xslm 等)。
- OLE 对象，以及相关特殊表格已经可以正常的兼容 MS Office 2007 版本格式。
- 文件加密功能已经支持对 MS Office 97/200/XP 格式的加密文件进行了全面支持。

当然一个大版本的更新内容肯定不会只有这么多，更多的详细更新还希望大家在使用过程中慢慢体会，小编认为一些新的改进的确不错（我就比较喜欢 Calc 中的表格多选功能）

不过 OpenOffice.org 目前也有一些短板，就是对 MS Office 的兼容性还是不能 100% 完美，对格式的支持看起来还是慢了一个版本，但是如果您对与 MS Office 的兼容性没有什么特殊要求的话，其实 OpenOffice.org 也不失为一个不错的选择，哦对了，更重要的一点，它是免费的！

Ubuntu 操作系统商业发展策略剖析

4 月 29 日准时发布的 Ubuntu 10.04 版再次点燃粉丝们对这款流行开源 Linux 系统的热情，而新版中更趋明显的商业化迹象也引发了争议。以笔者之见，Ubuntu 如今的成功（全球用户数达到 1200 万）实际上也是其背后商业发展策略正确的佐证。

梳理 Ubuntu 商业化发展策略，我们可以看到它是以 Linux 桌面版为发展重心，在桌面发行版成功的基础上，带动服务器版本进军企业级 Linux 市场，在这个过程中，通过各种衍生的增值服务（包括广告、在线音乐商店等）来拓展商业发展的空间。



这是一个非常聪明和灵活的策略，当别的厂商关注 Linux 高端应用的时候，Canonical (Ubuntu 的发行商) 选择从被认为是没有“钱”途的桌面版入手，从而避开与老牌 Linux 厂商的正面交锋。Canonical 深知，Ubuntu 发行版是商业化的基石，因此自身必须足够优秀。而这些年，

Canonical 在提高 Linux 的启动速度，增强系统的安全性，提高可用性和易用性上所做的努力也是

有目共睹的，这也保证了 Ubuntu 能很快从众多 Linux 发行版中脱颖而出。

而 ubuntu 在桌面市场的成功，反过来也带动了服务器市场的发展，有报道称，Ubuntu 服务器版已经成为继红帽之后，市场占有率第二的企业级 Linux 版本，这可以看作是 Canonical 以桌面市场包围服务器市场策略的成功。

从新版来看，令人印象深刻的启动速度，全新的外观以及长期支持策略等，为 Ubuntu 的正式商用奠定了坚实的基础。据悉，Canonical 已经与许多大的 PC 厂商如 Dell、联想等达成协议，将在他们的台式或笔记本电脑中预装 Ubuntu 10.04 版本，而 Ubuntu 10.04 版也将是 Ubuntu 大规模商用的正式开始。



作为企业级 Linux 的后来者，在 Ubuntu 发行版中加入对云计算技术的支持，也将是确保 Ubuntu 能在企业级 Linux 市场抢得先机的关键。Canonical 很早就提供了 Ubuntu One 云存储服务，并从 Ubuntu 9.10 版起在发行版中集成 Ubuntu One 的本地客户端程序。

在 Ubuntu 10.04 版中，音乐播放器 Rhythmbox 集成了“Ubuntu Music Store”（Ubuntu 音乐商店），用户可以付费购买喜欢的音乐。通过与大的 PC 厂商合作构建基于 Ubuntu 的云计算中心以及加强软、硬件在 Ubuntu 平台下的认证，都会促进 Ubuntu 在企业级 Linux 市场的发展。

然而，在 Ubuntu 商业化的过程中也会带来许多问题。例如在商业化过程中可能植入的一些第三方闭源程序会影响喜欢自由软件的用户。另外，为了商业利益，过于频繁的变化也会给用户带来不便，在 Ubuntu 10.04 版正式发布前的几个测试版中，发行商曾将 Firefox 浏览器中集成的 Google 搜索栏变成 Yahoo 搜索栏，但没过多久，浏览器的搜索栏又变回 Google 了，并且默认启动页面也指向 Google 的搜索引擎。其间缘由，不言自明。

Ubuntu 的商业化之路仍在探索前行中。对于用户而言，也许不会过多关注发行商的市场运作策略与行为，他们需要的只是简单易用、功能强大的产品，因此只要 Ubuntu 在商业化的进程中把握好商业化与用户需求之间的平衡，企业和用户的双赢局面就会一直持续。

新版 Ubuntu 五大看点

看点一 启动再次提速

在 Ubuntu 10.04 桌面版中，启动再次提速，在大多数计算机上都能以 20 秒的时间启动完毕，而更有网友经优化后将启动时间缩减到极短的 3.6 秒。这无疑让人对 Ubuntu 后续版本在启动速度方面的改进充满期待。

看点二 LTS（长期支持）版本

Ubuntu 10.04 是一个 LTS（Long term support，长期支持）版本，桌面版的支持时间为三年，而服务器版的支持时间则长达五年，这意味着在支持阶段，用户可以获得各种补丁等更新。

看点三 全新的主题与外观

Ubuntu 10.04 采用新的 Logo（标志），新的系统字体、新的启动闪屏画面、新的登录界面、新的主题、新的桌面背景以及采用 GNOME 集成桌面环境（2.30 版，采用了新的应用程序图标）等。Ubuntu 放弃了之前一直坚持的橙黄主题色，改以紫色为主，并且在设计时强调“Light”的效果。

看点四 集成最新的社区应用

Ubuntu 10.04 版集成微博客户端程序 Gwibber，它支持包括 Twitter、Jaiku、Facebook、Identi.ca 等在内的社区网络服务。使用 Gwibber 客户端工具，用户可以快捷、方便地与社区内的用户进行交流。

看点五 更好地支持云计算

Ubuntu 10.04 版对 Ubuntu One 统一在线存储服务的客户端进行了增强与改进，使得用户使用和管理云存储客户端工具更加便捷。另外，在 Ubuntu 服务器版中，集成了最新的私有云创建软件包，可以帮助用户以低廉的成本搭建单位内部的私有云。

细数 Office 2010 十大值得关注的新特性

1、更直观地表达想法

Office2010 开创了一些设计方法，让用户可以将想法生动地表达出来。使用新增的和改进的图片格式工具（例如，颜色饱和度和艺术效果）可以将文档画面转换为艺术品。在 Office2010 中，将这些工具与大量预置的新 Office 主题和 SmartArt 图形布局配合使用，可以更淋漓尽致地表达出自己的想法。

2、协作的绩效更高

在团队工作中，大家集思广益可以获得更好的解决方案并能更快地在限期内完成工作。当使用 Microsoft Word 2010、Microsoft PowerPoint 2010、Microsoft Excel Web App 和 Microsoft One Note Shared Notebooks 与其他人合作时，可以与他们同时处理一个文件，甚至可以身处各不相同的地方。



3、从更多地点更多设备上享受熟悉的 Office 体验

使用 Office 2010，您可以从更多地点更多设备上更轻松地完成任务。可以在智能手机或几乎每台连接至 Internet 的计算机上，随时随地进行工作。Microsoft Office Web Apps——将 Office 2010 体验扩展到 Web。可以在线存储 Word、Excel、PowerPoint 和 OneNote 文件，然后通过 Web 访问、查看、编辑和共享文件内容。Microsoft Office Mobile 2010——使用 Windows Mobile 智能手机专用的增强型移动版 Office 2010 应用程序，用户可以实时了解信息并快速回应。

4、提供强大的数据分析和可视化功能

使用 Excel 2010 中的数据分析和可视化功能可以跟踪和亮显重要的趋势。使用新增的 Sparklines 功能可以在工作表单元格中使用小图表来清晰简洁地表达数据。使用 Slicers 功能可以在多个层对 PivotTable 数据进行过滤和拆分，从而可以减少格式设置时间，增加分析时间。

5、创建出类拔萃的演示文稿

可以在演示文稿中使用个性化的视频来吸引观众。可以直接在 PowerPoint 2010 中插入和自定义视频，然后修剪、添加淡化方式和效果，或者在视频中标出关键点来引起观众对选定场景的注意。现在，插入的视频默认嵌入在文件中，这为用户省去了管理和发送额外视频文件的麻烦。

6、轻松管理大量电子邮件

可以将很长的电子邮件会话压缩成一些对话，并可对这些对话进行分类、存档、忽略和清除。使用新增的“快速步骤”功能，只需单击一下鼠标便可执行多个命令任务，例如回复并删除电子邮件，这样即节省时间又节约收件箱空间。

7、在一个位置存储并跟踪自己的所有想法和笔记

使用 OneNote 2010，即可在终极数字笔记本上跟踪、组织和共享文字、图片、视频和音频笔记。使用“版本跟踪”、“自动突出显示”和“链接笔记”等新功能，可以更好地管理笔记，从而始终能掌握自己的思路并能在团队协作过程中了解最新更改信息。

8、即时传递消息

可以将 PowerPoint 演示文稿广播给远程观众，不管他们有没有安装 PowerPoint。使用新增的“广播幻灯片”功能，可以通过 Web 浏览器快速分享演示文稿，无需执行其他任何设置。

9、更快、更轻松地完成任务

Microsoft Office Backstage 视图代替了传统的“文件”菜单，将所有文件管理任务（例如保存、共享、打印和发布功能）都集中到一个位置。所有 Office 2010 应用程序都增强了功能区界面，让用户可以快速访问命令，以及自定义选项卡实现个性化体验。

10、在不同的设备和平台上访问工作信息

您可以享受在更多地点更多设备上使用 Office 2010 的自由。使用 Microsoft Office 2010 时，无论您身在何处，均可以通过 PC、智能手机和 Web 浏览器来获得熟悉、直观的 Office 体验。

红旗软件成功通过 IBM WAS 6.0/7.0 认证

中科红旗 Asianux 的主要成员近日宣布，其产品 Asianux Server 3.0 已成功通过 IBM WAS 6.0/7.0 的官方认证。此次双方的携手合作，不仅会为用户提供更加牢固且简化的 IT 架构，还提供了高性能、节能化的绿色 IT 环境。同时，这一合作也为智慧城市、云计算、虚拟化等前沿技术架构，提供了一种高可用性、开放性和安全性的选择。

IBM 是 SOA 领域的市场领导者，拥有 8,000 多个客户，市场份额超过 53%

(Wintergreen)，也是位居第一的中间件产品市场领导厂商。IBM WebSphere Application Server（简称 WAS），即 IBM 的 WebSphere 应用服务器，是 IBM WebSphere 软件平台的基础和面向服务的体系结构的关键构件。WebSphere Application Server 提供了一个丰富的应用程序部署环境，其中具有全套的应用程序服务，包括用于事务管理、安全性、群集、性能、可用性、连接性和可伸缩性的功能。它与 Java EE 兼容，并为可与数据库交互并提供动态 Web 内容的 Java 组件、XML 和 Web 服务提供了可移植的 Web 部署平台。



Asianux Server 3.0 是由红旗软件公司牵头，联合亚洲主要国家（包括日本、韩国、泰国、越南等）联合开发的 Linux 服务器操作系统，其应用目标是面向亚洲企业系统的通用性 Linux 平台。它为企业级客户提供了业务运行所需要的高可靠性、扩展性、易管理性及更好的软硬件兼容性。

Asianux Server 3.0 通过 IBM WAS 6.0/7.0 的官方认证，不仅有力地保障了双方产品的兼容性，全面满足了客户系统需求，而且将为中国用户带来强大附加价值，为中国 Linux 用户带来完美的产品组合。Asianux Server 3.0 作为一款理想的操作平台，秉承 Linux 的开放性、可靠性和安全性，为 IBM WAS 用户带来更开放的平台选择，降低了对硬件平台的依赖性。另一方面，Asianux 卓越的 TCO（总体拥有成本）表现，为 IBM WAS 用户带来更低的总体拥有成本，也能极大地降低用户管理

和维护费用。IBM WAS 在 Asianux 平台的相关问题，都可以通过正规渠道及本地化支持得到完美解决，而 IBM WAS 平台在中间件市场的领导地位也将为 Asianux 用户带来企业级的产品选择。

红旗软件与 IBM 一直保持着良好的互利共赢的合作关系。之前，IBM 就扩大了对红旗软件产品的中间件、BladeCenter 和 System x 服务器的支持。在 2007 年，IBM 与红旗软件合作完成了 WebSphere, Lotus, Tivoli, System x, 和 BladeCenter 等全线产品的认证。此次红旗 Asianux Server 3.0 通过 IBM WAS 6.0/7.0 的官方认证，是 IBM 对红旗软件产品认证的一个标志性的新节点。双方将进一步加深合作，为用户提供灵活、可靠，绿色环保的解决方案。

公安部边防局技术处的相关负责人在应用该解决方案时认为：“基于红旗 Asianux 3.0 及 IBM WAS 7.0 的解决方案，在公安边防应用一体化系统中发挥出更加卓越的性能，特别是稳定性方面，表现优越。不仅帮助我们优化和规范业务流程，而且大大提高了我们的信息应用水平。”

IBM 软件集团大中华区 WebSphere 总经理韩菊芳表示：“IBM 公司坚持投资开放源代码市场，致力于对 Linux 的支持，为客户提供更多更好的选择。此次与红旗公司的合作，对 Asianux 的认证，就是践行了 IBM 公司在开源市场的承诺。凭借 Asianux 在中国 Linux 市场的领导地位，相信此次合作也会为 IBM WAS 带来更多市场机遇，也为使用 Asianux 及开源产品的用户，带来 IBM WAS 企业级的产品体验，移植更快捷、更平滑。”

红旗软件公司总裁兼 CEO 贾栋表示：“联合认证是在 IT 产业链形成过程中，厂商之间重要的互动环节，红旗软件作为 IBM 的重要商业合作伙伴，此次 Asianux 产品通过 IBM WAS 6.0/7.0 的认证，会为红旗产品用户带来更多的企业级产品选择，必将增强用户对红旗产品的使用信心，进而拓宽双方在更多领域的合作。”

MySQL 陷内忧外患已处于消亡的边缘？

甲骨文为了在收购 Sun 交易中获得 MySQL 费尽心思，才最终获得监管机构的批准，目前来看这些努力可能是在浪费时间和金钱，人们或将突然发现，内忧外患的 MySQL 已经处于消亡的边缘

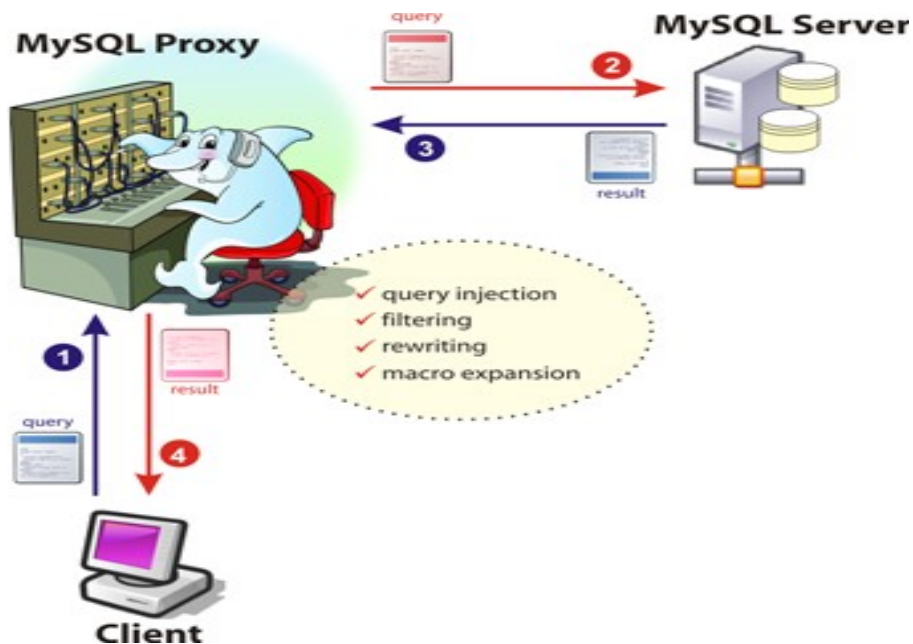
在上月举行的 MySQL 大会上，MySQL 之父迈克尔·韦德纽斯（Michael Widenius）和大名鼎鼎的 MySQL 架构师布莱恩·阿克（Brian Aker）分别发表演讲，他们坚信任何一家公司都不可能成为 MySQL 开发或支持服务的唯一提供商。这些 MySQL 名人的做法对甲骨文来说是一种考验，将验证甲骨文与 MySQL 社区配合和容忍不同意见的程度。

近日旧金山新创公司 Clustrix 公开宣称，自己的产品更强大更优秀，可以完成 MySQL 做不好的事情，可扩展至存储数十亿条数据，完全可以取代 MySQL。

Clustrix 产品中不存在 MySQL 的 DNA，但它可以与 MySQL 协议互通，这样应用程序再也无需进行代码移植，它的存在无疑会伤害 MySQL 的付费业务。

该产品被称为针对互联网规模级应用程序的首款集群数据库系统，据说它遵循了应用程序服务器和存储系统突变成可扩展式、群集产品的进化路线。

它具有 NoSQL 的 key/value 存储的巨大扩容能力和高性能，而且封装在 3 节点服务器 CLX 4010 设备内的 SQL 具有可靠的 ACID 测试相关功能，该硬件设备足以处理高负荷的读/写数据操作。



这三个或更多机架式设备都需要运行一个被称为 Sierra 集群数据库引擎的软件。据 Clustrix 称，用户希望或需要多大的可扩展性，取决于把多少节点设备加入到机架中。

Sierra 群集数据库引擎是一个非共享式执行环境，包含 Sierra 并行规划器（Parallel Planner）和 Sierra 分布式执行引擎（Distributed Execution Engine）。它把查询任务提供给分布式数据，而不是像 RDBMS 那样把数据提供给查询任务。

这意味着 Clustrix 群集数据库应该能够以最大的并行性执行查询语句，许多同步查询具有最大的并发性。这将带来极高的可扩展性、读/写操作性能、可用性、在线调整纲要、自我修复和自我管理。

Clustrix 团队从 Isilon Systems 那儿学到不少经验，后者层针对存储系统开发过类似的高并行和分布式产品。Clustrix 已经从风险投资机构那儿获得了 1800 万美元来研发可扩展数据库。

Clustrix 群集数据库的目标用户群是面向事务处理的云计算服务提供商、企业和社交网站类互联网公司，它们在处理互联网生活中令人难以置信的繁琐数据时，为了获取所需的扩展性，不得不忍受在应用程序层不断进行合库和拆库的操作。同样在解决该问题的还有开源项目 Hadoop 和 Cassandra，以及谷歌的 BigTable。

要想扩容 MySQL 数据库，通常需要许多令人痛苦的定制化编程，这是一个成本高且耗时的工作，而且在单实例数据库中很难找到互联网规模的关系数据库功能。Clustrix 承诺，借助于它的产品，人们不再需要这类代码编写工作。

Clustrix 的群集数据库系统能够以增量和无缝方式扩容至数百个节点，运行时就像一个单一实例数据库一样，具有全部关系数据库功能和一致的即时事务处理。

该工具可以被透明和不中断的部署到分片、非分片和复制 MySQL 环境中。当客户需要增加更多的 CLX 4010 节点时，这个分布式和并行体系架构可以自动发布数据到新的节点，即使在写数据负荷非常重的情况下，也能实现线性提高性能。它通过自动负载均衡、失效切换、还原和自我修复实现高可用性。

Clustrix 双核及四核设备包含两个 1Gbps 以太网口和两个 20Gbps 的 InfiniBand 背板端口，同时还装配 32GB RAM 和 7 个 160GB 固态硬盘。三节点设备的报价是 109995 美元。

Clustrix 表示，之前它已经开始销售这些产品，并且在今年第一季度达成第一笔交易。据称该公

司目前已经收到不少订单，很明显对某些 MySQL 客户来说，他们的产品比较有吸引力。

深入分析：Flash VS HTML5 到底谁统江山

Steve Jobs 在"Thoughts on Flash"一文中，谈及网络影音时，多次提到 HTML5 与 H.264 两项标准。但是，到底 HTML5 是什么？Flash 和 H.264 又是怎么回事？所以，本文旨在解释它们之间的关系，让大家可以初步了解。

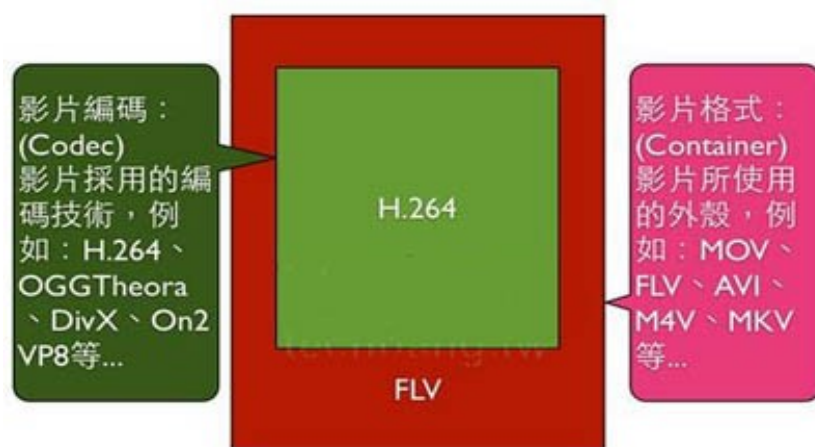
首先，引用 Steve Jobs 的一段话：

Adobe 一再反复宣称 Apple 的移动设备不能提供“完整的网络体验”，因为网络上 75% 的影片是 Flash 格式。但他们没有说这些影片几乎所有都属于一个更为先进的格式：H.264。

单这一段，已经令人不知道他到底在说什么，那先让我们来做个名词解释。Flash 影片，也就是我们常见的 FLV 格式视频，是一种常用的影片格式（Container），播放 FLV 时需要 Flash Player。而 H.264 是影片编码（Codec），适用于多种影片格式，像是 QuickTime 的 MOV 格式，Flash 的 FLV 格式。而目前在 Youku 等各大在线视频网站上的 FLV 影片，大多采用 H.264 作为编码，所以它们既是 Flash，又是 H.264。

了解两者之间的差异之后，现在开始说明 Flash 与 HTML5 在在线播放上的差别。首先，让我们谈谈电脑上的运作差异，以及稳定的纠结点。

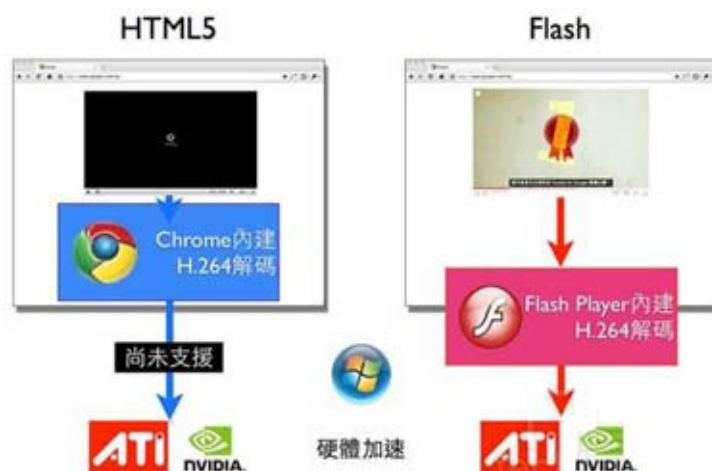
电脑浏览器上两者的差别在哪？又有哪些问题？



首先，让我们用 Windows 结合 Google Chrome 浏览器介绍播放 Youtube 视频时的运作模式：

1、Flash：Youtube 上的 FLV 视频需要通过一个 SWF（Shockwave Flash）播放器播放，而这个播放器会调用 Flash Player Plug-In 来播放影片。

2、HTML5：YouTube 上的 M4V 影片直接使用标签即可播放，而播放器是通过 JavaScript 编写的，一切都是利用浏览器内建功能完成。



图：YouTube 播放运作模式

这两种模式的优缺点，又可以用『稳定性』和『流畅性』来说明：

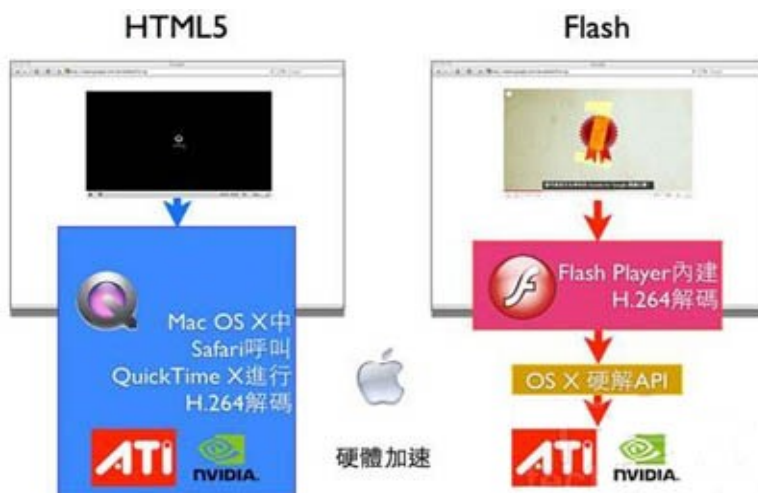
稳定性：

Flash Player 最大的问题是容易假死，想必大家都会在玩游戏，看电影时遇到 Flash Player 出现问题的情况。之前，只要 Flash Player 一假死，就会连浏览器都一起死掉。于是，Google Chrome 在推出的时候就一再强调每个程序都是独立的，这样一来，就算 Flash Player 假死，浏览器也不会收到牵连。之后的 Mac OS X 10.6 中的 Safari 也加入了这项功能，据说 Firefox3.6.4 也会加入。

但是，这样的功能只是避免浏览器随 Flash Player 陪葬而已，要是 Flash Player 稳定性不提高的话，问题最终还是没有解决。而 HTML5 利用标签和 JavaScript 来播放，不需要任何插件，这样相对稳定很多。

流畅度：

另一件事是，当在线视频由 480P 逐渐升级到 720P 或者 1080P 的时候，光是播放就已经占用相当高的 CPU 资源。台式机播放都有点吃力，更别提是 CULV 及 ATOM 的轻省本了。于是 Flash



图：HTML5 与 Flash 播放的不同

Player 10.1 版开始加入了硬解码支持，利用 GPU 加速来降低 CPU 资源的占用。实际测试也证明，

硬件加速是目前降低 CPU 占用的唯一办法。

在这项测试中，Mac 上的 Safari 播放 HTML5 影片时占用的资源相当少，主要是因为 Safari 遇到 H.264 格式的影片时，会调用 Mac OS X 中的 QuickTime X 进行解码，而 QuickTime X 原来就支持硬解。当时 Apple 尚未放出硬解 API，使得 Flash Player 10.0 与 10.1 完全没有差异。当 Apple 公布硬解 API 之后，Adobe 也随即推出 Flash Player 10.1 Gala For Mac，加入了硬解功能，CPU 占用也明显减少。微软也宣布将在 IE9 中支持 HTML5 影片播放和 H.264 硬解。

比较这两点，Flash Player 除了稳定性问题，支持跨平台（NO Linux）、跨浏览器硬解是比 HTML5 具有优势。而 HTML5 最大的问题在于浏览器内建编码不统一，这一点在下文再讨论，现在看看移动设备的状况。

移动设备：性能和电量很吃紧

Steve Jobs 在文中也提及，目前大多的移动设备芯片都支持 H.264 硬解，而目前移动设备主流芯片几乎都采用 ARM 架构。的确，ARM11、ARM Cortex-8、Cortex-9 大多都支持 H.264 硬解，但 Adobe 也宣称手机上的 Flash Player 10.1 将支持硬解。但是就算支持硬解，影片播放和硬件之间隔了一层 Flash Player 的话，电池消耗又会是如何呢？依照 FlashMobileBlog 的实测，采用 WiFi 上网，Nexus One 能看 3 个小时的 Youtube 视频，但 Nexus One 官方数据中，单机播放影片的时间达 7 小时。出去无线连接耗用的电量，与 Steve Jobs 所提及的 iPhone 可播放 10 小时的 H.264 影片，但播放 Flash 影片仅有 5 小时，比例大概是 2: 1，可见 Flash 耗电确实是个问题。无论如何，再过几天，Adobe 将于 Google I/O 开发者大会上展示。

供 Android 使用的 Flash Player 10.1，也极有可能随着 Android 2.2 版（Froyo）一起更新，到时候就可以见分晓。



不过，如果 Adobe 不能实际证明 Flash 在移动设备上不耗电的话，通过 HTML5 直接在线播放 H.264 影片则是目前移动设备最佳的解决方案。

HTML5 影片的纠结点：浏览器支持解码不一

編碼格式	Opera	Mozilla Firefox	Google Chrome	Apple Safari	MS IE9
OGG					
H.264					

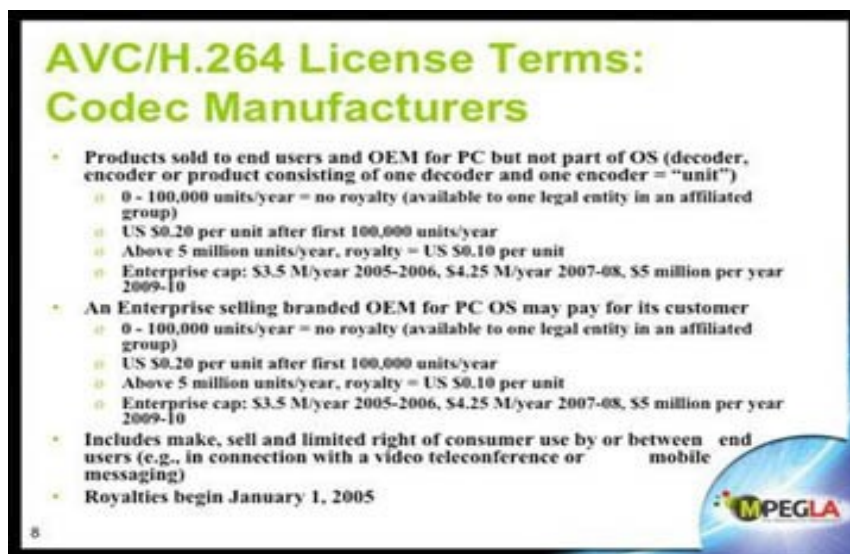
图：主流浏览器对视频编码的支持

Steve Jobs 发表公开信后，一名欧洲自由软件协会（FSFE，Free Software Foundation Europe）的实习生 Hugo Roy 写了一封公开信，认为 H.264 不是真正的开源标准，因为未符合五项该协会的定义。Steve Jobs 回信表示：

所有的影片编码都有着专利。Ogg Thora 与其他“开放源码”的编码目前背后有着一个正在组织中的专利联盟（注一）。不幸地，我们不能因为某件事物是开放源码，就代表或承认它并未侵犯其他人的专利。开放标准并不等同于免授权费或开放原始码。

到底为什么 H.264 不被 Mozilla 与 Opera 采用，这个还需要进一步说明。

1、H.264 又称作 MPEG 4 Part 10，和 MPEG 2 一样需要授权费，并统一由 MPEGLA 这个专利联盟管理收取。使用 MPEG 4 标准比编码/解码都需要付授权费，一年产品总算在 10 万个以后时免费，但超过 10 万个的时候，每个产品将收 0.2 美金的授权费，超过 500 万时，授权费降为 0.1 美金，上限则是 500 万美金。



2、线上免费内容，如 YouTube 等视频网站可免费使用到 2016 年。但如果提供租借电影，像 NetFix，就要依照用户数量收取授权费；如果用于 PPV（Pay Per View）以及 VOD（Video on Deman），像是 MOD 与 BBTv 数码有线电视上的收费电影，超过 12 分钟的内容，也要收取售价的 2% 授权费。最多以 500 万为上限。

因为这样的授权费用规定，支持 GPL 协议的 Mozilla 基金会，以及支持开源标准的 Opera，都力挺开放源代码的 OGG Theora 格式，而未内建 H.264 编码。所以尽管它们都支持 HTML5 的标签，却无法播放 H.264 格式的视频。

HTML5 影片在移动设备上，除了微软的 Windows Mobile 上的 IE 不支持 HTML5 之外，其他采

用 WebKit 核心的浏览器的手机大多支持 H.264 编码，成为通过格式问题不大。但回到电脑上，毕竟 Opera 是商业公司，不排除未来会内建 H.264 的可能；但 Mozilla 基金会就完全变成推上的钉子户，占浏览器 30% 的 Firefox 使用者，就是看不到 H.264 格式的视频。

而 Google，将会是打开这个僵局的重要角色。Google 去年收购了影片编码研发公司 On2，并且传言将会在 Google I/O 大会上，将旗下的 VP8 编码开放成为开放原始码。如此一来，具有接近 H.264 编码低流量、高品质，又开源的 VP8，将有可能成为统一 HTML5 影音的一匹黑马。

结论：

1、移动设备上：Flash 如果耗电问题没有解决，HTML5 与 H.264 硬件将会是未来较好的在线播放格式。

2、PC 终端：如果主流浏览器没有统一支持 HTML5 编码，普及性将会无法和 Flash 比

3、Apple：以苹果产品线的齐全及软硬件的配合，在产品上推广 HTML5 不难。但要扩展出去有困难。

注一：微软过去曾将 WMP9 编码开放源代码，结果导致许多公司要求编码的权利。于是成立 VC-1 这样的专利联盟（Patents Pool），一共有 16 家公司组成，包括微软。

Novell 和 Red Hat 在用户界面专利案中获胜

据国外媒体报道，美国联邦陪审团近日表示，Novell 和 Red Hat 两家公司并没有侵犯由 IP Innovation LLC 公司持有的用户界面专利。

位于伊利诺斯州的 IP Innovation LLC 公司从 2007 年开始指控上述两家公司，声称他们侵犯了其三项专利。在这些专利中，最主要的是发表于 1991 年的美国专利，专利号为 No. 5, 072, 412，专利为一项在不同的计算机之间共享工作平台的技术。根据 IP Innovation LLC 公司的指控，Red Hat 的 Linux 系统和 Novell 的 Suse Linux Enterprise Desktop 以及 Novell 为 System z 构建的 SUSE Linux Enterprise Server 等都使用了这项技术。

但是，美国德克萨斯州东区地方法院的陪审团发现，IP Innovation LLC 公司并没有提供足够的证据来支持他们的指控，因此驳回了对 Novell 和 Red Hat 的判决。

Red Hat 的执行副总裁迈克尔·坎宁安（Michael Cunningham）在一份声明中说：“陪审团发现这是三项无效的专利，但是被伪装成重要的和新的专利。”

据悉，IP Innovation LLC 公司的律师没有立即就此事发表意见。

开源漫谈之 GNU GPL 的前世今生

由于参与开源运动的多为开发者与设计者，而开源许可协议的原文往往跟法律条文一样难懂，导致开源社区的很多参与者并没有对开源许可协议产生足够的认识。但事实上，了解开源许可协议不仅可以帮助我们更好的做出选择，并且可以更加深入的参与开源社区的文化。

要了解开源许可协议，当然要从 GNU GPL 开始说起。这个许可协议究竟是如何诞生，又是如何被社区接受并广为使用的呢？请看下面的介绍。

GNU GPL 条文概述

GNU General Public License，通常简称为 GPL，是当下最为通用的开源许可协议。GPL 的条文最早在 1989 年由 Richard Stallman (RMS) 撰写，用于 GNU 项目。这位自由软件之父在当年提出了四大自由的诉求：

- ◆可以使用软件做任何事的自由
- ◆可以根据自己需要任意修改软件的自由
- ◆可以与别人分享软件的自由
- ◆可以与别人分享自己对软件所做改动的自由



为了有效地保护这些自由，RMS 将 GPL 定义为第一个、同时也是最为严格的 Copyleft 许可协议——也就是说，如果你使用了按照 GPL 发布的项目，那么你的项目也必须按照 GPL 许可协议发布，不得添加任何其他限制。（不过，Copyleft 这个理念却不是从 GPL 开始，而是从 20 世纪 70 年代起就有人开始实行的。）使用 GPL 协议的代码制作的程序并非不能拿来卖钱，事实上 RMS 开始自由运动的第一笔经费就是通过售卖附在磁碟上的 Emacs 赚来的。GPL 的主要规定在于源代码必须自由公开。之后为了增强此协议的 Copyleft 效力以及与专有软件代码以及其他协议代码的兼容性，RMS 又先后在 1991 年和 2007 年敲定了 GPLv2 和 GPLv3 的条文。

使用 GPL 的知名开源项目

根据 Black Duck Open Source Resource Center（直译为黑鸭子开源资源中心，由 Black Duck 软件公司发起建立并维护的数据库）的调查，目前使用 GNU GPLv2 的开源项目是最多的。到今天为止（2010 年 5 月）的统计中，GPLv2 占据了全部开源项目当中 48.54% 的比例，而 GPLv3 的使用率也已经不低，达到了 5.61%。这其中包括很多我们所熟悉的开源项目：

Linux：如日中天的操作系统，和 Unix 以及整个自由软件运动有着非常深的渊源。目前使用 GPLv2（因此在理论上，所有的 Linux 发行版都遵循 GPL 许可协议）。

GNU 系列：GNU Emacs，GNU 调试器，GNU C 编译器等等。当然，GNU 项目已经在使用 GPLv3。

eMule 电驴：众所周知的 P2P 下载工具，目前使用 GPLv2。有兴趣的读者可以看看他们翻译的非官方 GPL 中文文本。

WordPress：流行的博客系统，目前使用 GPLv2。

Java：Java 的 HotSpot 技术和 Java 语言编译器 javac 从 2006 年开始采用 GPLv2，不过条款中在 GPLv2 允许的范围内使用了 linking exception（直译为连接例外），所以是一个修改过的

GPLv2。另外，同属于 Sun 的 Glassfish 和 NetBeans IDE 也都采用此种许可协议。



MySQL：著名的开源数据库。MySQL 采用的是双重许可协议——带有例外的 GPLv2，以及专有类型的最终用户许可协议。这种双重许可的方式是开源软件发展多年的一个产物，这种方式允许厂商通过售卖开源软件的许可证赚钱，为想要使用 GPL 开源项目而又不想公开自己代码的用户提供了另一种选择。至于这种方法是否有利于开源界，以及是否会被用来钻空子，目前还无法得出明确的结论。不过可以肯定的一点是，因为 MySQL 在 GPLv2 下发布，所以无论之后发生什么事，MySQL 这个开源技术都是不可能被扼杀的。

GPL 牵涉的法律案件回顾

须知 RMS 所做的远不止是开发了 GNU 项目以及确定了 GNU GPL 许可协议。为了自由软件的推广，RMS 在 1985 年创建了自由软件基金会（Free Software Foundation，FSF），并一直致力于解决自由软件运动相关的法律和结构问题。

可想而知，GPL 在推出之后并开始流行的十多年间成为了很多专有软件厂商的眼中钉。最早的一起直接针对 GPL 的法律诉讼发生在 2003 年 8 月，身为原告方的 SCO Group 以 Linux 内核中使用了 IBM AIX 代码，而 AIX 代码又使用了 SCO Unix 代码为由，将 IBM 告上了法庭。这一举动激发了 Linux 社区的公愤，IBM 继而联手红帽针对 SCO 提起反诉与上诉；之后 SCO 开始恐吓不愿撤销 SCO Unix 使用权的 Linux 用户，并更进一步展开了对 Novell、AutoZone 和 DaimlerChrysler 的诉讼。这一系列诉讼逐渐演变成了一场旷日持久的战争。

这场战争的战场之一就在于 GPL：根据 SCO 的申诉，SCO Unix 的源代码是当时 SCO 的一些员工私自按照 GPL 发布的，因此不具备法律效力。SCO 的发言人还进一步表示，“GPL 违背美国宪法，还有版权，反托拉斯，及出口管制等法令。”但是，整个事情还要更加复杂：Unix 原本是 AT&T 卖给 Novell 而 Novell 又卖给 SCO 的，（这段历史可参考 Unix 传奇一文）因此自由软件和开源社区反过来质疑 SCO 是否真正拥有 Unix 的所有权；而 FSF 也站出来，依据 GPL 的条文捍卫 Linux 用户不受恐吓的权利。

美国法院最终并没有在 GPL 许可协议上进行定论，但数年间的法律判决基本都对 SCO 不利，2005 年的判决中甚至对 SCO 没有提供侵权代码的证据表示了直白的鄙夷。SCO 最终在 2007 年申请了破产保护，并希望卖出自己的 Unix 产品线，但这似乎也不可能了：美国法院在那一年宣判 Unix 属于 Novell，并在最近的 2010 年 3 月 31 日重复了这个判决。虽然这场战争的主战场与 GPL 关系不大，但 Linux 一方的胜利还是为 GPL 增添了一定力量。在 2007 年，一个 FLOSS 一方的叫做 SFLC（Software Freedom Law Center，软件自由法律中心）的法律团体以违反 GPL 协议为由，代表 BusyBox 的两位开发者将 Monsoon Multimedia 软件厂商告上了法庭。在此之前，针对违反

GPL 协议的处理方法一直是由 FSF 等开源组织私下沟通解决的，因此此案成为了美国第一例因 GPL 而立案的案件。不过，这宗诉讼最后以双方庭外和解的方式结束，GPL 也因此失去了一次验证其法律有效性的机会。



为什么社区对 GPL 如此青睐？

GPL 是一个代表了 RMS 的 Copyleft 理念的许可协议，在各种开源许可协议中属于最为严格的一个，可以说是直接阻碍了开发者通过贩卖版权和专利软件这种传统的软件赚钱营生。但是为什么在这么多的开源项目当中，使用 GPL 协议的项目占据了一半以上的高比例呢？由于开发者大多数专注于开发本身，而对许可协议这个概念并不十分重视，所以有些项目的发起人在没有进行深入了解的时候直接选取了流行的 GPL 协议，这是原因之一。不过有很多社区人士则明确的表达过自己选择 GPL 的理由——

“GPL 代表了自由。”

“我使用 GPL，因为微软讨厌它。我不用 BSD，因为微软喜欢它。”

“GPL 意味着分享，而 BSD/MIT 协议意味着偷窃。当然，这取决于你如何定义分享与偷窃！”

众多观点之中，Linux 之父 Linus Torvalds 的说明毫无疑问是值得了解的。在 2008 年的一次访谈中，Linus 对许可协议的选择是这样评论的：

“我相信 GPL（尤其是 v2）是一个协同工作的绝佳模式——所有的人分享他们的代码，同时确保没有浑水摸鱼者能够利用他人的工作成果来为自己取得好处——你使用自己贡献的源代码来为你所获得的源代码进行“支付”。我将其称之为 tit-for-tat 模式，这种模式不仅适用于软件界，也同样在经济学与博弈论中十分出名。不过这种 tit-for-tat 模式并非所有人的需求。比如说，如果你隶属标准委员会，你只是希望能够借由一段开源代码的传播来推广一个标准，而并不介意这段代码是否会被用在专有软件中盈利，那么 Apache 或 BSD 协议则比较合适。

即使单纯是从理性的角度来看，不同许可协议的存在也都是合理的。同时我也要说，程序员们

并非在任何时刻都是理性的。自负的心理，个人的特殊需求，都是导致众多有细微差别的许可协议诞生的原因。

不过我们要知道，有选择是件好事！而且非常流行的许可协议并不是那么多，所以基本上没有什么可困扰的。”



讲到这里，相信大家对于 GNU GPL 开源许可协议的由来和使用情况已经有了相当的了解。当然正如同上面介绍的，在开源运动的发展历程中，GPL 并非是最早诞生的许可协议，而且随着时间的推移与各种需求的影响，社区中出现了很多其他的开源许可协议，其中也有不少是由 GPL 修改而来的。这些其他的许可协议是如何诞生，它们和 GPL 有什么不同，而社区又为什么会选择这些许可协议呢？

资深 Linux 系统使用者的玩具清单

目前开源 Linux 系统无法广泛的推广主要还是和用户对相关软件的使用情况不了解。很多人担心不能使用自己习惯的软件，这个问题存在于很多人的心理。作为一个资深的 Linux 系统用户，我从 2005 年 4 月开始使用 Linux，到去年 8 月时全面从 Windows 转到 Linux 环境，至今所经历的时间大约也要满两年了。在这期间，我尝试了许多的 Linux 软件，经过大浪淘沙，有些软件最终成为经典被我一直沿用下来。我打算建立一个清单，一来可以为自己留存备忘，二来也可为他人聊作一点参考。

首先从 Linux 发行版说起。我目前使用的是 Ubuntu。个人认为这是一个比较易用的 Linux 发行版。我曾经也使用过 RedHat 的 Fedora 和 Debian，但时间并不长。我对 Arch Linux、Gentoo 等也比较感兴趣，在未来不排除有使用它们的可能。

桌面环境或窗口管理器：迄今为止，我使用时间最长的桌面环境是 GNOME。有一段时间，我不忍 GNOME 的速度，使用 Fluxbox 了很长时间。另外，Xfce 也曾短暂使用过。我还没有深入使用 KDE，也许要等到我的新机器到手之时。现在，我依然回到了 GNOME，其足够简洁的界面已可让我合意。

办公套件：OpenOffice.org 这套软件包括字处理、电子表、演示稿等组件，我在工作中时常

使用它们。

图形处理：我曾经使用扫描仪将一些个人照片保存到电脑中，在编辑它们时，我会用到 GIMP。我也使用 Inkscape 和 Xara 做一些小的设计。

网络浏览：Firefox 是我首选的网络浏览器，我对于它的使用几乎从未间断过。我知道 Opera 也是一个比较好的选择，很轻量，速度也不错。但使用 Firefox 似乎已成为了一种习惯。

听歌看影：最初听歌时，我用的是 XMMS，它经典而古老。后来，无论听，还是管理，都离不开 Quod Libet 的身影。至于看电影的选择，我一直都在用 Mplayer，它支持的格式真的很广泛。

游戏娱乐：也许是曾经很久积淀下来的街机情结，我通常会玩一些模拟游戏。这时候，就需要 Xmame 了。系统中自带的一些扑克牌游戏，我也会玩一玩。

文本编辑器：在没有遇到 Scribes 之前，我使用 Vim 来满足日常的文本编辑需要。而现在，多数时候都在用 Scribes，它的智能、以及它的灵活，个人都十分喜欢。我也渴望用一用 Emacs，我甚至买了一本有关它的书，但还需要一段时间的学习，才能熟悉它。

FTP 工具：之前有用过 gFTP 和 FileZilla。但这两个工具时不时会出现一些小问题，如 gFTP 在删除大量文件时会崩溃，而 FileZilla 自前一段时间出现“检索目录列表失败”的问题后一直无法正常使用。最终我选择了 lftp，感觉基于命令行的它使用起来更高效些。

聊天：原来是使用 Gaim，它支持许多的协议，我的 Gtalk、MSN、QQ 帐号都可以登录。现在，我在用 Gajim，使用 Gtalk 帐号登录与朋友们进行交流。

浏览图片：gThumb 和 GQview 在系统中都装上了的，但 gThumb 还是要用的多一些。

电子书阅读：PDF 用的是 Evince，一切以够用为主；CHM 之前用 xCHM，现在用 KchmViewer，后者对于多语言、多编码的支持要好得多。

邮件收发：基于对 Firefox 的好感，自然地选择了 Thunderbird。有时候，也直接就在 Gmail 中进行处理。

文件管理：使用 GNOME 默认的文件管理器 Nautilus。也会用到 ROX-Filer，这个速度要快一些，而且也很方便灵活。

终端程序：我使用终端程序的过程是这样一条路线：GNOME Terminal->Xterm->rxvt-unicode->Tilda。

RSS 阅读：我一直都在坚持使用 Liferea，从 1.0 到现在的 1.2，一些不错的功能也逐渐开始添加了进来。

虚拟机：我使用 VMware 有很长的时间了，包括当初还未转换到 Linux 之时。后来，我也使用过 QEMU，但其 100%CPU 的占用率，让我不得不换掉它。现在，我以为 VirtualBox 是一个很好的选择。

作笔记：FreeMind 可以助我达成此目的。看书时、逛网时，思考时，有用的东西都可以利用 FreeMind 记录下来，以供日后查阅。

下载：BT 下载方面对于我来说，µTorrent 可能是最合意的了，虽然它是一个 Win 程序。不过，最近听 Dark 说起它可能会出一个 Linux 版本，倒是很让人期待。另外，aMule 也是一个很好的工具，我通常会使用它下载一些电影或是音乐。

其实，在使用 Linux 的这段时间中，我接触到了大量的软件，其中有很多都十分优秀，限于篇幅，恕我不能一一列出。以上所列出的囿于个人的见识，也难免挂一漏万。

如果你也在使用 Linux，不妨也来列一列这个清单。我相信这不仅是一种良好的交流，也是一种有益的补充。

更新

Linux 发行版：从 2007 年 11 月开始，我使用 Arch Linux。

音乐播放：MPD+Sonata 是我目前的选择。

图片浏览：我觉得，Mirage 是一个更适合我的图片浏览软件。

MeeGo 步入互联网战局

在不久的将来，智能手机操作系统除 Symbian、WM、Android、iPhoneOS、OPhoneOS 等以外，还将迎来新的挑战者 MeeGo。这是英特尔 Moblin 和诺基亚 Maemo 两大操作系统融合后的产品，基于 Linux 系统的软件开发平台，向应用开发者提供了通用计算架构上的统一操作环境。

面对 Google、苹果在智能手机市场站稳脚跟后，向平板电脑等手持设备蚕食的势头，牢牢控制着电脑平台的英特尔也感受到了来自移动互联网的商机与挑战。英特尔全球副总裁兼中国区总裁杨叙宣称，MeeGo 的出现将帮助产业链围绕个性化互联网走向更广泛融合。“软件是充分发挥所有个性化互联网设备潜能的关键要素，英特尔有一整套关于软件战略的构想。”

MeeGo 一亮相，就成为 4 月 13 日开幕的 IDF2010（英特尔信息技术峰会）上的热点。与苹果、谷歌等类似，英特尔也在采取垂直整合的方式渗透整个产业链。英特尔希望以诺基亚与自己的强大影响力，借助 MeeGo 系统，将 Moorestown 平台、未来的 AppUp Center 软件商店都推向前台，当然还包括功耗更低的 Atom 凌动处理器。

发力个人移动终端

来自摩根斯坦利的互联网分析师 Mary Meeker 称全球已经进入以移动互联技术的普及为标志的第五个重大科技周期，集中体现为“3G+社交网络+视频+网络电话+强大的移动设备”。Mary Meeker 预计未来五年移动互联网将超过桌面互联网。

巨大的市场空间和发展机遇，英特尔自然不会置之不理。杨叙在与记者沟通时表示，从现在开始要发力个人移动终端市场。“三网融合最大的受益者不是终端设备商，而是内容和服务这一块，今后如果想有更广的操作系统平台，就要让更多的人将自己开发的东西放到网上供人下载，然后使用到各个屏幕和设备上，都能够兼容，这也是英特尔要做的。”

在杨叙看来，三网融合或物联网都属于“个性化互联网时代”，而这一时代正在启动，他希望英特尔能与合作伙伴一起迎接这一机遇。因此，在 IDF 上，英特尔将智能电视、智能汽车、数字标牌、数字家居以及节能房屋等“搬上”演讲台，以多元化方式为参会者营造了三网融合及物联网全新体验，包括 MeeGo 的演示。

显然，英特尔希望能在這個新時代里繼續成為全球 ICT 產業的領導者之一。在經過 2009 年的一系列專業實驗之後，英特爾開始將 MeeGo 開源軟件平台正式向應用開發者和消費者推出。儘管是與諾基亞的戰略合作，但是 MeeGo 却是要“支持不同硬件架構的廣泛設備種類”，包括下一代智能手機、上網本、平板電腦、媒體電話、聯網電視機和車載信息娛樂系統等。其一網打盡移動互聯

网时代终端领域的庞大野心，显而易见。

一个值得关注的细节是，此次英特尔信息技术峰会上公开的 MeeGo 1.0 的系统界面，并非大家所希望的智能手机，而是以 Acer 的 Aspire One 上网本为介质来做的演示。从画面上看，它继承了更多 Moblin 的特色，多任务、大量软件、快速启动条、3D 游戏，以及 SNS 社交应用。

据悉，MeeGo 的首个版本将在 2010 年第二季度发布，而 MeeGo 设备将在 2010 年 6 月亮相。届时，是否是诺基亚新推出的智能手机，目前还不得而知。不过，根据 IDC 数据预测，2010 年智能手机市场将超过 2 亿部，全部手机市场有望超过 13 亿部。

有人说，只要基于 Linux 的 MeeGo 操作系统能占领市场，那么，英特尔架构的各种处理器就能顺理成章地抢占新的市场，无论是智能手机还是互联网电视，都将是英特尔的天下，改变英特尔在这些新领域受 ARM 压制的局面。

或许，这才是英特尔的真正目的。事实上，三网融合的各种技术创新都源于服务的创新，企业要想在这个领域获得成功，就必须认识到，今后所有应用都一定要从垂直整合的角度来进行设计，从最底层的芯片到操作系统，到如何搭建应用软件，再到如何转化为服务。

“傍大牌”大步向前

MeeGo 一出生，就在其核心操作系统软件库提供了包括 MeeGo 操作系统、中间件等资源，可供开源社区下载、开发以及测试。因此，MeeGo 不仅是一个操作平台，更是一种新的商业模式。

作为一种新的平台或概念，产业链更关心 MeeGo 能为上下游厂商带来何种价值。英特尔全球移动互联网总监 Pankaj Kedia 曾说 MeeGo 是一个“赚钱魔方”：“MeeGo 可以跨平台使用，包括 PC、Symbian 和 MeeGo 上。开发商开发的同一款软件将可以赚三次钱，用户开发的软件也可以放在多个软件平台上销售。此外，开发者开发的软件除了可以在诺基亚的 OVI 软件商店上销售之外，还可以放在英特尔和合作商的软件商场里，这样开发商可以更快变现。”据悉，英特尔已经拥有 3000 万名开发者。

记者在 IDF 位于国家会议中心 4 楼的 MeeGo 展台区里看到，几乎所有的 Linux 厂商都来到了这里，展示着各自基于英特尔 MeeGo 系统的终端产品。“之前我们应用英特尔的 Moblin 技术，在其基础上客户优化了嵌入式 Linux 软件和针对上网本的开源操作系统，在部分神舟电脑、惠普等品牌的上网本中预装，这几天还来不及把宣传单改为‘MeeGo’。”中科红旗展台的技术人员告诉记者。

除了开源厂商，更多的嵌入式设备和解决方案供应商是 MeeGo 依靠的另一个大牌。4 月 14 日，作为一个实例，英特尔在现场邀请了华泰汽车展示其尚未对外公布的新款轿车“华泰元田 B11”，华泰汽车集团副总经理王殿明现场演示了车内采用英特尔凌动处理器和 MeeGo 的车载信息系统，“未来，用户可随时下载服务提醒和维护更新等信息，并使用在线音乐、即时天气预报和导航等娱乐应用。”

英特尔公司高级副总裁兼软件与服务事业部总经理詹睿妮说，MeeGo 希望有更多的合作伙伴。“但我们会专注于少数几个联合合作的项目，要与那些有共同理想、共同愿景，有前瞻的移动互联网共同向前。”在英特尔的“大牌”合作者名单上，除了华泰汽车外，还有腾讯、东软集团、中科红旗、搜狐无线、优视科技、凯立德等各个行业的领先者。4 月 13 日，英特尔还特别与腾讯签署了合作开发 MeeGo 软件平台的意向书，双方将携手开拓集通信、互动及娱乐于一体的移动应用。据悉，东软集团已建立起 MeeGo 能力中心，不久就会有一系列基于 MeeGo 的车载、消费电子等产品上市。

“把第三方的开发者移到 MeeGo 平台上开发应用软件，是 MeeGo 能否成功的关键。”开源

软件推进联盟主席陆首群认为，开源软件供应商多数还跋涉在通往大规模盈利的征途上，英特尔可以拿出一些资源来培养这个群体，“和自己一起跳舞。”

FLOW 让你提前尝试 Google 的云操作系统

Google 的 Chrome OS 预计将在年底前正式推出，但 Google 的粉丝们不想等待这么久。他们开始尝试开源开发者自制的 Chrome OS 版本。《纽约时报》采访了 17 岁的 FLOW 开发者 Hexxeh。

Hexxeh 的真名叫 Liam McLoughlin，是英国曼彻斯特人，一位大学生和程序员，他下载了 Chromium 源代码，编译出一个可通过 U 盘启动电脑的镜像，他花了无数夜晚和周末配置 Chromium，使其能在不同类似电脑上运行，包括苹果电脑，他还添加了 Google 还没有加入的功能，如 Java 支持。McLoughlin 称他在 Chromium 上的工作一部分是为了展示他的计算机技能，另一部分它也可能是打开通往技术行业的一扇大门。对 Google 来说，所有围绕 Chrome 替代的开发活动都有可能成为一把双刃剑，它希望开发者和企业能在官方版本上开发，Google 没有预料到，在 Chrome OS 还没准备好前人们就开始尝试和评估 Chromium 了。负责产品管理的副总裁 Sundar Pichai 虽然认为 Chromium 的提前发布是一个始料未及的结果，但承认，如果你开始做一个开源项目，你就必须开放所有道路。

IBM 收购云计算软件商 Cast Iron

IBM 宣布，已收购了私人控股的软件公司 Cast Iron Systems，以求进一步增强自己的云计算技术能力。云计算是一种日益流行的电脑技术，借助在线访问软件来帮助企业削减 IT 成本。今天 IBM 未透露收购协议的细节。IBM 指出，本次交易将帮助客户整合来自 Salesforce.com、亚马逊、NetSuite 和 SAP 等不同软件提供商的各类云计算应用程序。

过去的十年中，逐渐远离硬件业务集中精力拓展软件及服务使 IBM 获益不浅。今天 IBM 还称，预计至 2012 年时，全球云计算市场的总额将由 2008 年的 470 亿美元增至 1260 亿美元。

Eclipse 宣布新的 SOA 平台启动

根据 Eclipse 基金会的官方报道，其宣布一个新的 SOA 产业工作组成立。这项新计划的目标是定义一个通用的 Equinox 为基础的 SOA 平台，包括工具和 Runtime 组件，可以由供应商，系统集成和 SOA 部署的企业使用。



该工作组包括 Engineering Group，itemis，Obeo 和 Sopera 等。这些公司均将使用 Eclipse SOA 平台，作为自己的商业解决方案。此次合作重点在于定义“Eclipse 的 SOA 套件”，其将在 eclipse.org 提供下载。

Eclipse 的 SOA 平台将包括一些关键技术领域，例如：

- * REST 和 SOAP 对 OSGi 远程服务的支持。

*一个建立在开放标准上的可扩展的 SOA 框架，它提供企业级功能，如 Runtime 服务注册与基于政策的端点协商与消息处理的整合。

*一个新的扩展业务活动监控（eBAM）项目，该项目将提供一个平台来进行外部系统的性能监控和管理分析。

*一个新的 Eclipse 业务流程管理（eBPM）项目，将提供一个完整的 BPM 解决方案（工具和 Runtime），建立在 Equinox 和 OSGi 基础上。

*工具的支持，以帮助开发人员方便创建，测试和部署的 JAX-WS Web 服务。

现在，Eclipse SOA 的第一个版本已经提供下载，2010 年 6 月，一个更新版本会发布。

赛门铁克：Linux 垃圾邮件威胁严重

据国外媒体报道，赛门铁克公司旗下在线信息和网络安全服务商 MessageLabs 最新的研究报告称，相比其他操作系统，Linux 垃圾邮件数量与其市场份额极不相称。



通过被动特征探测（Passive Fingerprinting）方法，MessageLabs 监测了 2009 年 11 月份至 2010 年 3 月份间各类操作系统的垃圾邮件传播情况。

MessageLabs 分析师 Paul Wood（保罗-伍德）表示，Windows 系统占据着 90%以上的市场

SpamIndex (%Spam / MarketShare)	
Windows	1.01
Linux	4.99
MacOS	-
Other	1.08

Figure 18 - Likelihood of a computer running

份额，因此绝大部分的垃圾邮件仍然来自 Windows 系统。

据统计，92.65%的垃圾邮件来自 Windows PC，2.22%的垃圾邮件来自其他系统，而苹果系统几乎没有任何垃圾邮件。值得注意的是，5.14%的垃圾邮件来自市场份额仅为 1.03%的 Linux 系统。

OS Market Share	
Windows	91.58%
MacOS	5.33%
Linux	1.03%
Other	2.06%

Figure 17 - Market share

MessageLabs 还统计了每款系统的垃圾邮件传播比率。结果显示，Linux 的传播比率最高，为 4.99。Windows 传播比率仅为 1.01，苹果系统为 0，其他系统为 1.0。

OS % of Spam	
Windows	92.65%
Linux	5.14%
Other	2.22%
MacOS	0.00%

Figure 16 - Proportion

伍德表示：“Windows 系统应该是最容易被用于传播垃圾邮件，但 Linux 系统的统计结果出乎我们的意料。”

社区扫描

Ubuntu 10.04 LTS 新功能阐释

Ubuntu 10.04 LTS 的最大卖点是启动速度有了极大的提升。据官方介绍是可以在上网本上实现 10 秒内完全启动。这个卖点可是相当的诱人，毕竟现今上网本的弊病是速度过慢。其实除了这些，Ubuntu 10.04 LTS 在娱乐办公上有了很大的改进，下面让我们来看看到底 Ubuntu 10.04 LTS 到底有哪些新的功能吧。



全新的 Ubuntu 10.04 LTS

网页浏览：Ubuntu 自带了 Firefox 浏览器，让你享受更快、更安全的网页浏览。同时，你也可以在 Ubuntu 软件中心下载其他的开源浏览器，如 Chrome，Opera 等。

办公应用：OpenOffice 完美兼容微软 Office 文件，帮你实现在 Ubuntu 上 Word 创建与编辑，PowerPoint 创建与编辑，Excel 表格创建与编辑等操作。

海量免费软件：Ubuntu 软件中心拥有数千种免费开源软件。你可以找到网络应用软件、游戏、音乐和视频、编程和办公应用等。这些软件更加容易安装和卸载

即时聊天：Ubuntu 可以更加方便地绑定你的聊天账户，支持绑定 Yahoo，Gtalk，MSN，Jabber，AOL 甚至是 QQ 哦。Evolution 可以快速绑定你的个人邮箱，提供更加直观易用的邮件服务。

社区网络：Ubuntu 10.04 可以快速绑定你的社区网络账户，及时收取更新你的社区网络信息。不知道在国内能否支持新浪微博和开心网的绑定呢？国外提供的是脸谱（FaceBook）和推特（Twitter）绑定。

音乐管理：Ubuntu 10.04 内置全新的音乐管理库，让你可以更加方便直观的管理的你音乐数据。更加方便地将你的音乐分享给你的朋友。支持现有主流音频格式。

相片管理：Ubuntu 将会是你相片管理的好帮手，它可以很好地帮你管理无论是相机、手机还是网络下载的各种相片。可以更加简单的分享到 Picasa，面书，Flickr 上（貌似在国内都被河蟹掉了）。当然，你也可以在 Ubuntu 软件中心下载免费的图片编辑软件。

资料存储：每个 Ubuntu 用户都将获得一个免费的 Ubuntu 账户。它可以在线存储你的书签、音乐和图片。随时随地都可以调用这些资料。

视频管理：Ubuntu 可以让你更加快速的观看在线视频，编辑管理本地视频。

快速启动：Ubuntu 10.04 LTS 仅需几秒钟的时间即可完全启动，让可以更加快速的进入工作或者娱乐。

上千种免费游戏：在 Ubuntu 软件中心，我们可以找到上千种免费游戏。你只需下载安装即可进行游戏，而不需要支付任何费用。

桌面操作：Ubuntu10.04 LTS 将提供一个全新的桌面操作系统。让你可以体验到易用、华丽、流畅、免费的操作体验。

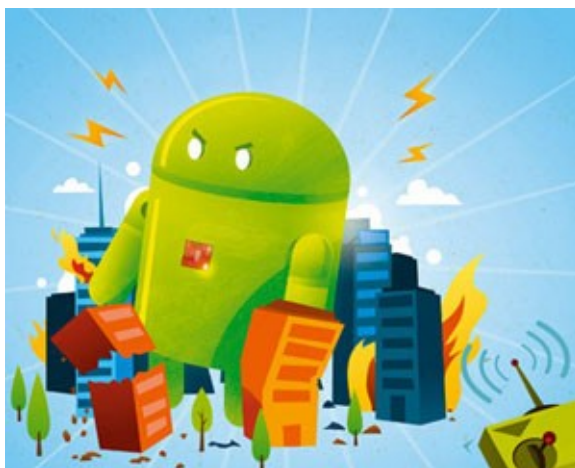
最封闭的开源系统：话说 Android 的八宗罪

你以为 Android 是开放的吗？Google 采用了一系列的控制手段来保证每一部 Android 手机上都有它指定的软件和硬件规格。然而，他们同时又利用 Android SDK 里面的 Apache 许可证来大肆鼓吹 Android 是开放的。

没错，Google 的移动平台是当前最聪明的利用开源来驱动商业议程的实现。但在我们深入探讨这个为什么之前，我们先说说为什么 Android 的成功和开源并没有什么关系吧。

是什么成就了 Android

虽然早期饱受质疑，Google 的 Android 移动平台已经在移动行业得到了营运商和手机厂商的广泛支持，仅剩固执的诺基亚。Android 从 08 年的一款机型发展到 10 年的 50 多款，发展之快让绝大多数的业内观察家们吃惊。



Android 的成功和开源毫无关系。它的成功依靠下列三个主要因素：

-苹果. 这点看起来很奇怪，Android 竟然是靠它的主要对手发家的？让我来分析下。在 iPhone 空前绝后的成功以及苹果对网络营运商傲慢苛刻的态度下，营运商们迫切的在寻找一种更便宜的选择。因此这些第一层最大的营运商们开始积极的用 Android 来开发手机给那些买不起 iPhone 的用户，更重要的是，他们不需要每卖一部手机就给 Apple 300 欧元以上的回扣。

-全世界的营运商们迫切希望自己鹤立鸡群。Android 给他们提供了一个统一的软件平台。他们可以很方便的定制自己想要的系统，而且花费的代价也很低（3 个月的时间，这个比 SavaJe12 个月以上的定制周期要短很多）。对大型的营运商来说，Android 也降低了他们在智能手机软件方面的投资。这也是为什么大多数的 Android 手机项目背后都是营运商和 OEM 厂商的组合。

-高通。这个市值 100 亿美元的芯片厂商对 Android 的崛起功不可没。手机开发产商可以直接拿高通已经为 Android 集成好的方案，在 9-12 个月的时间内向市场上推广。（相比起来摩托罗拉的 CLIQ 花了 16 个月，而 HTC G1 则花费了 2 年多的时间）。除了高通，我们还有 TI 的 OMAP3 平台（摩托罗拉 Droid/Milestone 基于此方案）。ST Ericsson 和 Broadcom 也在做 Android 的集成方

案。

换句话说，在 Android 手机上，大多数的 OEM 预算花在了定制方面。而 Symbian 的绝大部分预算花在无线通信的移植和硬件整合上了（Symbian 2001 年所做决定的结果）。总的来说，Android 使 OEM 厂商可以大幅削减研发预算，把钱花在定制这个刀刃上。当然我们不能忘记 Android 是免费的。这个免费让众多厂商激动不已。

话说回来，Android 用开源来做市场宣传，非常成功的搅乱了整个行业，导致了诺基亚对 Symbian 的收购以及 Windows Mobile 的全面崩溃（不过译者觉得 iPhone OS 4 的多重任务机制的发布让 WP7 真正成了杯具帝）。不过更重要的是，利用开源的名号和 Google 的魅力，Android 吸引了成千上万的开发者，虽然 Android 并不能让开发者们赚到很多钱，而且 Android 手机的数量不到苹果产品的十分之一（连支持收费的国家都比苹果少 6 倍）。

在开源的面纱后面

让人更惊讶的是 Android 到底有多封闭，尽管外面包裹着 Google “不作恶”的口号和 Apache 授权许可证模式。借用亨利福特在 Model-T 相关的书里的一句话：“任何人都可以自由挑选 Android 的颜色，只要那是黑色”（anyone can have Android in their own colour as long as it's black）。Android 是一个绝好的商业案例——展现一家公司是如何用开源来赢得关注和社区参与，而且同时保持一个非常严密的商业运作。

Google 是如何控制着每台 Android 手机里采用什么服务、软件和硬件的？这个搜索巨人建立了一套很完善的控制管理系统。为了挖掘更多的信息，我们花了两个月，和很多与 Android 有着紧密联系的内部人士进行了讨论。我们发掘出的事实让人震惊。从宏观方面说，Google 控制 Android 手机构成以下八宗罪：

1、私有分支。Android 有多个私有分支，这些只给几个特定合作伙伴，往往是那些开发 Android 的 OEM 厂商，而且这些只提供给需要知道的人。这些私有分支比已经公布的 SDK 要超前起码 6 个月，也是 OEM 厂商可以保持竞争力的关键。而公开的 SDK 则是为第三方应用提供私有分支里发布的最新功能。

2、封闭的评估流程。所有的代码评估员似乎都是 Google 员工，也就是说从社区提交的代码只有 Google 才有权力决定是否接受。而且 Google 内部还流传着“并非此处发明”的一种思考文化，他们觉得 Google 员工写的代码是天下无敌。随便问任何一个给 Android 提交过补丁的人，你会得到一样的答复：几乎没有什么提交被 Google 接受，而被拒绝的时候往往没有任何理由和解释。

3、进化的速度。Google 对 Android 的创新的移动行业内绝无仅有的，他们在 18 个月里发布了四个大版本。想在 Android 上面做文章的 OEM 厂商只得紧跟 Google 的步伐（这里想起了移动杯具的 OMS），不然就跟不上新功能的发布和 bug 修复。Nexus One、Droid、G1 和其它带有 Google 体验应用的手机给 Google 提供了创新的测试场。

4、不完善的软件。用公开的 SDK 并不能完整的建造手机。缺少的几个关键的部份包括无线通信的集成模块、国际化语言包、营运商信息包以及闭源的 Google 应用，比如 Market、Gmail 和 Gtalk。虽然 Cyanogen 可以自己定制 ROM，但里面包含的那些应用没有授权，所以不能发布在商业用途的 Android 手机上。

5、闭门的开发者社区。Android Market 是唯一一个拥有超过四万个程序并和每个手机 OEM 厂商都签有合约的 android 程序商店。这个限制很要命，因为没有一个是 OEM 厂商愿意发布没有 Market 的 Android 手机（天朝是另类）。当然，在 Market 上发布应用是个非常简单的事情，没有

什么审批的步骤，这个和苹果的 AppStore 刚好相反。

6、反分化合约。外界几乎不清楚原来 OHA 的成员都签署了反分化的合约。但这个合约更可以被理解为不能发布没有通过 CTS 兼容测试的手机。（下面细说 CTS）（译者注：貌似中国移动已经被踢出 Android 的私有分支，是不是因为他们建立的 OMS 违反了 this 协定呢？）

7、保密的发展蓝图。Android 的发展蓝图是很杯具的，到目前为止，公开发布的发展蓝图还停留在 2009 年的第一季度。如果想要看到内部的发展蓝图，你需要 Google 的赐福。

8、Android 商标。Google 掌握着 Android 的注册商标和冠名权。任何想用 Android 品牌的厂商都需要得到 Google 的授权。简单的说：进 Google 的门，或者没有门。如果你要自己做 Android 分枝，你就全部靠自己了，比如你需要中国移动那么大的公司。

Android 的传奇中还有个篇章：CTS（兼容测试组），也就是 Google 一套测试 Android 手机是不是达到 Google 的标准。根据我们的线人消息，CTS 不仅仅测试软件的 API 部份，它还包括性能测试，硬件功能，设备设计，UI 用户界面需求，和机内打包的服务。CTS 决定了你可以添加额外功能，但不能从最基础的配置中削减功能。除了 CTS 以外，OEM 厂商还要和 Google 签订授权合同，这样他们才能打包 Google 的服务，比如 Gmail、YouTube 等等。

CTS 限制了 OEM 定制弱化版 Android 手机的想法（译者注：山寨的机会啊！MTK、中微星，年底发布些低端 Android 手机吧！）。这也大大限制了 Android 开拓低端市场的能力。CTS 和向前兼容 4 万多个应用的事实，极大的挑战着 Google 想占领智能手机市场 2 位数的市场份额目标。这些限制，还有 Google 与 OEM 亦敌亦友的合作关系，使得 OEM 圈内掀起了建立 Android 基金的讨论。

Google 的终极目标

手握 Android，Google 的目标是为自己产生收入的服务提供一个稳定的平台。在当前，这个广告生意。但未来，Google 的目标在语音服务（几十亿没有数据服务的用户）和 Google Checkout（比如变成移动领域的 visa 卡）。但不管 Google 的终极目标是什么，我们应该意识到 Android 和 Windows Mobile、Mac OSX 或 PalmOS 相比，并没有开放多少。Android 是用开源来驱动商业议程的最聪明的案例之一。Android 骨子里并没有我们潜意识里所灌输的那么多不作恶思想。

尘埃落定 法院判定 Novell 拥有 Unix 版权

美国犹他州地方法院对 SCO 集团和 Novell 之间关于 Unix 版权的问题做出了裁决，判定 Novell 拥有 Unix 的版权，而不是 SCO 集团。Novell 随即发表官方声明，证实了法院的判决结果。

Novell 对于法院的判决十分高兴：“对于法院认定 Novell 拥有 Unix 版权我们感到由衷欣慰，SCO 之前曾坚持 Unix 是他们的，而且以此来攻击 Linux。Novell 今后将继续致力于推广 Linux，包括捍卫 Linux 的知识产权。”

Novell 总裁兼首席执行官 Ron Hovsepian 表示：“这一决定对于 Novell、Linux 以及开源社区来说都是个好消息，我们一直认为（SCO 的）这种对抗 Linux 的举动是没有任何依据的，我们很高兴见到法官一致认同了这一看法。对于 Novell 在捍卫 Linux 和开源社区权利的行动中起到的作用，我感到十分欣慰。”

这起 Unix 所有权官司始于 1995 年，那时 Novell 将 Unix 的原始码及其他数据卖给 SCO 的前身 Santa Cruz Operation，允许 SCO 接管 Unix 提供服务技术服务。SCO 认为 Unix 版权归自己所有，

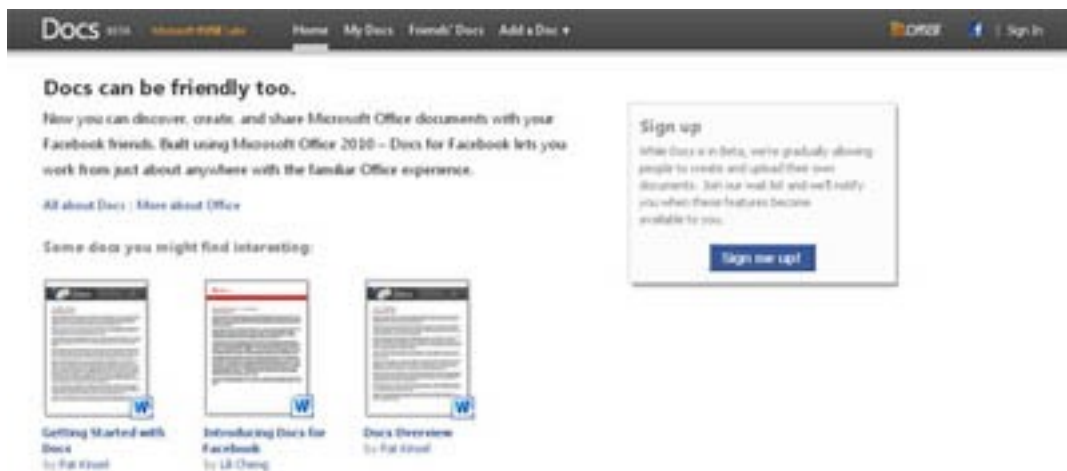
然而 Novell 则表示当年仅出售了服务权，并未出售 Unix 版权。

SCO 曾于 2003 年指控 IBM 非法将 Unix 技术捐赠给 Linux 开源社区，因此开源社区对此案也十分关注，如果 Novell 败诉，SCO 会转而指控其他 Unix 或 Linux 业者。因此，法院的判决对于开源社区来说是一颗定心丸。

Facebook 推出在线文档共享网站 Docs. com

4 月 22 日凌晨消息，Facebook 在 f8 开发者大会上宣布，与微软联合推出在线文档共享网站 Docs.com，与谷歌的 Google Docs 服务类似，今天已经正式上线。

据悉，Docs.com 允许 Facebook 用户使用 Facebook Connect 登陆，并可以创建、修改微软 Office 文档，以及与 Facebook 上的好友进行共享。新的文档将在用户的源（feed）中更新，就像用户状态更新一样。



Docs.com 网站首页

目前该网站正在处于测试阶段。微软计划于今年晚些时候推出自身的文档共享服务网站，Docs.com 正好可以为其提前进行功能测试。

分析称 Docs.com 可能不太会用于商业，因为本身需要基于 Facebook 的用户好友实现共享，这对于大公司来说不太容易实现，对一些小的公司或组织更可行一些。

Oracle 收购 Sun 并不总是坏事的 8 大原因

关于 Oracle 为什么要收购 Sun 已经谈论了很久，人们不禁会问，自由世界真会消亡，同时产生一个垄断的数据库世界吗？虽然回答的语气可能有点肯定，但我们也应该记住，这起收购对最终用户应该是有利的。

1、开源数据库已经站稳脚跟

无论 Oracle 是放弃，改进或使 MySQL 保持原样，都不要紧，重要的是其它开源数据库已经获得越来越多的认可，以 PostgreSQL 为例，自 Oracle 宣布收购 Sun 后，PostgreSQL 数据库和迁移工具的下下载量就暴增，这也证明了 Oracle 对开源市场的影响无关紧要。

2、NoSQL 有机会出头了

虽然关系数据和 NoSQL 处于不同的竞争环境，Oracle 收购 Sun 也促使许多组织开始寻找 MySQL 的替代品，并重新思考关系数据库是否真的适合他们的应用，是否有另一种选择。以前的中小型企业总是盲目地安装 MySQL，而现在的企业都很理性地看待数据库选型了，如 Twitter, Rackspace, Digg, Facebook, Cisco, Mahalo 和 Ooyala 都开始转移到 NoSQL 平台。

3、Java 会继续受到重用

不用我多说，Oracle 已经表态“Java 是计算机行业最知名的，应用最广的技术，它是 Oracle 收购的最重要的软件，Oracle 业务增长最快的融合中间件就是建立在 Sun 的 Java 语言之上的，Oracle 将会继续加大对 Java 的创新和投入，造福于客户和 Java 社区”。很多人质疑 Sun 被收购一事，主要是担心 Java 和 MySQL 的前途，很明显，如果 Oracle 是一台赚钱的机器，必然会打压 MySQL，进而促进 Oracle 数据库的销售，但一定不会打压 Java，因为那是 Oracle 的根基，还有谣言盛传如果 Oracle 将 Java 技术进一步壮大，将来不排除收费的可能。

4、开源社区应该受益

不管别人说什么，我不认为 Oracle 是开源杀手，Oracle 一直在参与开源社区，甚至比 Sun 还参与得更多，现在 MySQL 和 Java 都到了 Oracle 的口袋，Oracle 应该继续支持和促进这些项目的创新，Oracle 有这个能力，唯一的问题是 Oracle 在这方面会在多大程度上帮助 MySQL 和 Java，我们也许可以预计 Oracle 会限制 MySQL，但绝不会限制 Java，因为 Java 可以帮助 Oracle 紧密地集成软件和硬件产品。

5、Solaris 不会死去

有人认为，如果 Oracle 没有收购 Sun，Sun 可能会搁置 Solaris 操作系统，Oracle 获得 Solaris 后，终于可以继续挖掘和发展 Solaris 独特的高端功能了，Solaris 不但不会就此死去，还可能会继续发扬光大。

6、Oracle 将提供一个完整的解决方案

Oracle 收购 Sun 对 Oracle 现在的客户明显有利，因为 Oracle 现在有能力提供一个结合硬件，操作系统，数据库，中间件和应用程序的完整解决方案，在降低总体拥有成本的同时，在性能、可靠性和安全性等方面得到进一步改善。

7、我们将得到一个精干的供应商

许多人对 Oracle 大规模裁员感到不满，但我们不要忘记，在收购之前，Sun 已经开始裁员，虽然我不喜欢裁员，但它确实可以产生积极影响。作为一个消费者，你真的想先与硬件厂商交涉，再与软件厂商交涉？我个人更喜欢与系统集成商交涉，这正是 Oracle 和 Sun 合作伙伴即将得到的结果，因为以后他们只需要与一个单一的供应商打交道了，也将获得一致的支持。Sun 有 1.1 万渠道合作伙伴，他们将成为 Oracle 产品线的推动新生力量，Oracle 也有 2.1 万渠道合作伙伴，他们将成为 Sun 产品推动的新生力量。此外，我们也看到了有些项目被关闭或取消了，如外部人员将不能访问 Kenai 项目，也不再有 Sun Cloud 公共云服务。可以看出，Oracle 希望拿出双方过硬的产品参与竞争，重新树立一种精干的企业形象。

8、知识整合

虽然在不久的将来，Oracle 的技术网络和 Sun 的 BigAdmin 及开发者网络将会独立，同样

Oracle 认证培训体系将得到加强，将包括 MySQL，Java，SPARC 和 Solaris，当我想到在数据库培训中插入系统培训，或在系统培训中插入数据库培训，我就浑身起鸡皮疙瘩，但如果我们采用了 Oracle 的一站式解决方案，这不正是我们需要的吗？

让我们面对现实吧，我个人认为 Oracle 是收购 Sun 最恰当的公司，如果是其它公司收购了 Sun，我们要讨论的问题又不一样了，试想，如果其它公司拥有了 Java 控制权会是什么样子，MySQL 和 Solaris 会受到什么待遇？从一个数据库从业者的角度，我更愿意看到现在的样子。

Linux 与 Mac OS 赛跑，谁领先？

大家知道，Linux 与 Mac OS 的发展思路很不一样，根本坐不到一块儿议事，平日各走各的路，相互不搭理。但是，Phoronix 实验室非要把它们拉进实验室进行比试一番，并非出自它们的两厢情愿。情况是怎样的呢？

Phoronix 实验室把 Ubuntu 10.04 (Beta 1，第一测试版) 作为 Linux 的代表，以其与 Mac OS X 10.6.2 (最新稳定版) 进行比试，3 月 30 日公布了测试结果。在 20 个很复杂的测试项目中，9 项几乎打成平局，Mac 险胜 7 局，Ubuntu 大胜 4 局。所谓“险胜”是指超越量不很大，而“大胜”是指超越量较大、甚至很大。但是，Mac OS 有一项具有绝对的优势，压倒了 Ubuntu。结果，Phoronix 实验室宣布 Mac OS 获胜 (in general)，但是，最后又加上一句话，“This though could all change around again by the time Ubuntu 10.10 is released late into the year “。



为什么，Phoronix 实验室要这么说呢？

实际情况是，Ubuntu 10.04 使用了 Linux 2.6.32 最新内核，而且采用 EXT4 文件系统，如此以来，与 PostgreSQL 数据库有点儿“冲突”（需要多次反复读盘），非常耽误时间，在这项比赛中，自然 Ubuntu 比不过 Mac OS。但是，到了 Ubuntu 10.10 版本发布（解决来上述“冲突”）之后，情况就很可能大不一样了。

上述实验说明了一个事实，在 Linux 和 Mac OS 赛跑过程中，选手 Ubuntu 10.04 (Beta 1) 的表现不错，紧紧跟着对手不松劲，很有可能，Mac OS 的领先只是暂时的。实际上，我们要知道，Ubuntu 10.04 版本主要推荐使用 MySQL 数据库，不多涉及 PostgreSQL 数据库，对我们的实际使用影响不大。

在此，顺便说一下 Ubuntu 10.10 版本。近 2~3 个月以来，在 Ubuntu 国际社区里面，对于 10.04 版本的版面设计议论纷纷，说“Ubuntu 决策不民主”，把窗口放大、缩小和关闭的按钮放在左上角，而不放在传统的右上角，为此争吵得很厉害。3 月 30 日，Ubuntu 开发团队终于透露一个小秘密：右上角是预留给 10.10 版本用的，那里将会安置一个重要的新功能按钮。

记得在两年之前，Ubuntu 奠基人 Mark Shuttleworth 曾经说这样的话，在计算机桌面创新方面，Ubuntu 要领跑。现在看来，事实确实如此。这两年来，我自己一直在使用 Ubuntu 计算机桌面进行写作，现在使用的就是 10.04 (Beta 1) 版本。根据我自己的切身体验，Ubuntu 始终在不断地进步和演变。

Google 在扼杀开源贡献者？

很多优秀的开发者在进入 Google 之前都是非常活跃的开源贡献者，但是进入 Google 之后往往就销声匿迹了，包括嘲笑了此现象的 Memcached 作者 Brad 在进入 Google 之后也无法逃脱此规律。Brad 在最近一篇文章 *Contributing to Open Source projects* 谈到相关原因。

- * 许多优秀开发者都很喜欢编程，他们喜欢研究有趣有挑战的问题，并不特别在意这些项目是否开源。

- * 大家都太忙，Google 似乎用尽了每个人的空余时间。并不是说 Google 强迫大家一天到晚都在干活，而是由于 Google 里面太多有趣的东西做了，Brad 经常挂在口头一句话就是“现在手头有 7 个属于 20% 空余时间的项目”。

- * Google 的开发环境太好了，源代码控制，build 系统，code review 工具，debugger 调试工具，profiler 调优工具，submit queues，continuous builds，test bots，文档以及所有相关的自动化工具及流程非常完善。因此很容易 hack 任何项目，在任何地方，或者给任何人提交 patch，并且值得一提的是，很容易找到对应的人或者 list 去提交 patch。通常说来，提交 patch 是参与特性讨论，表达诚意的最好方式，即使你的 patch 是有问题的。

从上面尤其是第 3 点来看，Google 确实是技术人员的理想环境。

白宫向开源项目捐赠代码

美国总统奥巴马的官邸——白宫宣布向开源项目捐赠代码。

白宫网站采用了开源的 Drupal 内容管理系统，白宫技术人员对 Drupal 进行了改进，公布了修改后的代码。这些代码在可扩展性、通信和可访问性三个方面增加了 Drupal 的功能。在可扩展性方面，增加了一个叫 Context HTTP Headers 和 “Akamai” 的模块，前者可告诉服务器如何处理特定网页，后者则是让网站与内容发行网络 Akamai 整合起来；管理照片和视频内容的新模块 Node Embed；整合 CMS 和政府邮件系统的模块 GovDelivery。

行业观察

MySQL 数据库研发团队女掌门加盟 EnterpriseDB

加盟开放源代码数据库公司 EnterpriseDB，负责公司旗下 Postgres Plus 技术和产品营销团队。在甲骨文今年年初收购 Sun 之后，包括 MySQL 创始人在内的不少高层选择离开，Karen Tegan Padir 就是其中一位。EnterpriseDB 研发以开放源代码数据库平台 PostgreSQL 为基础的产品。Karen Tegan Padir 表示，当务之急是要提升 PostgreSQL 的吸引力，公司将加大对 PostgreSQL 的支持力度。

目前，MySQL、PostgreSQL、EnterpriseDB 是全球三大开源数据库。MySQL 被称为是最受欢迎的开源数据库，如今其前途因为甲骨文与 Sun 并购而前途未卜，众多反对者担心 MySQL 可能遭到 Oracle 的抛弃甚至打压。



不过，甲骨文首席架构师 Edward Screven 日前表示，公司计划增加对 MySQL 的投资，以进一步完善 MySQL 的功能，但其没有透露具体投资金额。

RIA 之战 微软欲借开源策略后来居上？

近 10 年以来，人们一直在试图寻找一个更好的 Web 开发的解决方案。最初是 HTML 和 CSS，后来才有了 AJAX 和 Web 2.0。但是因为 HTML 模型是基于页面的模型，缺少客户端智能机制，所以到目前为止基于 HTML 的 Web 应用程序对完成复杂应用方面始终跟不上步伐，整体的用户体验效果与桌面应用程序仍然有差距。

微软和 Adobe 作为 Web 应用领域的两大巨头，主要是采用为 Web 应用程序植入插件的方式来巩固 Web 战略。近日，在美国旧金山举行的开源大会（Open Source Business Conference）上，微软在“Web 为平台”专家讨论会上宣布了针对社区群体的 RIA 新开源策略，将公开其 RIA 技

术 Silverlight 的源代码。难道是巨人转性，还是另有隐情？开源策略在微软 RIA 战役中扮演什么样的角色？

一、开源 RIA 将引导微软走向开源时代

(1) RIA 将成为互联网的主流

在互联网诞生的时候，大家都在琢磨怎么把信息通过网络主动发布出去，出于共享信息的简单目的，一种快速小型超文本语言（HTML）被创建了。历经了多次的修改和完善，众多与 Internet 相关的技术纷纷出现，从 DHTML、XML 到 Java Applet、SWT、AJAX、Flash 等，这些技术有些是平行发展，有些是一脉相承。但毫无例外的都追求着一个共同的目标，就是更加强大、更高效反应、更加灵敏和更精彩的可视化特性的互联网程序。

也就是说，在过去几年中，Web 开发人员一直想构建一种比传统 HTML 更丰富的客户端，要实现比用 HTML 实现的接口更加健壮、反应更加灵敏和更具有令人感兴趣的可视化特性。这时，RIA 技术出现了，它允许我们在互联网上以一种像使用 Web 一样简单的方式来部署富客户端程序。RIA 是一种互联网应用程序，RIA 目前在很多地方既指富互联网应用系统，又指富互联网应用系统的开发技术，我们也可以将其理解为下一代互联网的应用程序。

那么，RIA 将来会成为互联网的主流么？这是一个只有一个答案的问题，那就是“会”。因此，一场新的技术战争已经悄然在 RIA 领域打响了。毫无疑问，Adobe 的 Flash 是 RIA 技术领域中最具优势的选手，但是微软借助 Silverlight 技术正在改变这种情况。例如，Silverlight 可以运行在所有 Web 浏览器上，而不仅仅是微软的 IE 浏览器；而且 Silverlight 还采用了打破微软多年老规矩的开源策略。



(2) 微软开放 Silverlight 源码，向开源社区示好

一直对开源吝啬、不感冒的微软居然也一反常态地对开源社区示好，公开其 RIA 技术 Silverlight 的源代码，是巨人转性还是另有隐情呢？这当然不能仅仅只用巨人转性来解释，也不是仅仅是因为 Adobe 宣称将开放其用于 Flex RIA 环境下的软件开发包代码的回应。据有关专家分析，微软公布

Silverlight 技术的部分源代码，以此表示对开源组织的友善态度，目的是为了藉此吸引开发社区的关注，更好地同 Adobe 展开竞争，更是其欲称霸 Web 平台领域的一种新姿态。因为居于 RIA 开发工具领先地位的 Adobe，也正希望借开放其 Flex 部分源码来巩固优势，而作为挑战者的微软想要赶上 Adobe，当然也必须要借助开源来吸引开发者使用 Silverlight。

微软此举从策略上来讲，可称为是实现在网络领域的一次飞跃。当然，开源并没有微软以前想的那么可怕，让 Silverlight 开源反而将更有好处：它将极大地扩展 Silverlight 的市场，以最快的速度普及。当 Silverlight 足够普及时，微软必定会看到更多的新利益。实际上，微软的许多对手都是携着开源的力量来与微软竞争的，如 Google 携开源势力强势入侵微软的多个传统强势领域，开源的 Firefox 又在不断抢占微软 IE 的市场份额，而这次 Adobe Flex RIA 也打起了开源的主意，微软终于坐不住了，也破天荒地宣布 Silverlight 的开源策略。对于未来的展望，我们有理由相信只要微软尝到了 Silverlight 开源的甜头，微软就可能会尝试更多的开源计划，这也可能会成为使微软加入开源阵营的一个起点。

二、为什么微软 RIA 需要采用开源策略？

在 RIA 市场中，选用哪家的工具，一直不是最关键议题，重点是在于开发者是在哪一个 RIA 生态环境中。因为为了确保 RIA 可与内部的核心系统相联结，一个认同某个 RIA 生态环境的企业决策主管，极可能会下达或建议开发人员选择哪家的开发 RIA 工具。在这样的状况下，可以预见对 RIA 生态环境的认识和习惯将会成为 Web 开发决策的关键因素。

据 Forrester 公司的分析师表示，在 RIA 市场上两大巨头 Adobe 和微软各具优势。Flash 先入为主，目前已经有了一个很大的市场，Flash 技术已经应用于 90%左右的 PC 上。微软想要拉拢这些真正的市场推动者则要花上不小的力气，而微软在 RIA 技术 Silverlight 上应用开源策略正是出于拉拢这些市场推动者的考虑。那么，为什么微软 Silverlight 的发布与推广需要采用开源策略呢？

(1) Web 开发者需要培训

从用户体验的角度来说，我们甚至还没有发挥出 HTML 的全部潜力。因此，大部分开发者认为基于标准的 Web 开发还大有潜力可挖，通过加强设计者与开发者的联系，Web 产品的用户体验可以得到很大的提升。目前大部分 Web 应用还不能令人满意，但是这个问题并不完全归咎于浏览器的非标准实现，更多的是因为设计和开发者本身对 Web 技术的掌握还不到位。

一般来说，Web 设计者往往感性些，通常是半技术性人员，喜欢定期购买和升级软件产品。而开发者则更理性，是纯技术性人员，卖给他们工具非常困难。和多数工程师一样，开发者通常更喜欢自己创建工具，或者使用免费提供的开源工具。开发者之所以更接受开源，是因为开源可以让他们控制自己使用的工具。因此，微软只有通过开源的方式才有望追赶 Adobe。

(2) 做大 RIA 市场规模，需要更广泛的协作

Adobe 已推出多年的 Flash 技术与 Flex 工具是公认比较成熟的 RIA 解决方案，拥有绝大部分的市场占有率优势；而微软推出的 Silverlight 技术是挟程序开发领域的优势，进入网页应用市场。但为了让 RIA 市场更为蓬勃发展，则需要建立起更广泛的 RIA 生态体系，这意味着微软不能只依赖设计人员或开发人员等单一族群。所以，通过开源策略可让 RIA 的开发人员和设计人员迅速理解 RIA 内部运行机制。因为就技术人员来说，开源可以极大的提高人员的技术水平，通过对开源的学习可以以最快的速度对很多基本的东西加以理解。

另外，开源技术的灵活性可使得它能够比专有解决方案更易于添加更多自由和个性化的功能。这是因为开源技术的开发、测试和发布过程完全是透明的，同时提供的源代码及完善的文档，有助

于开发者清楚地了解开源技术的工作原理和实现方法，也更容易得到质量更好的实现方案。这就保证了开源技术除功能上不逊于封闭源代码的方案外，还具有更高的灵活性，以及更低的采购和使用成本。因此，开源能对整个 RIA 开发生态环境的技术发展起到极大的推动作用。

三、为什么开源策略是微软决胜的关键？

为什么微软要在 RIA 技术 Silverlight 上开源？对于这个问题，很大一部分人的观点是认为微软终于抵挡不住开源社区和竞争对手的种种压力，最终被迫开放源代码。但实情却非如此简单，实际上开源策略是对微软的未来有着深远的影响和战略性的意义。

(1) 开发社群活跃度成生存的关键

市场就是市场，大鱼吃小鱼的商场不是只认技术是否先进的，IT 技术市场上的博弈使任何一个决策失误后果都被放大。技术折腾不过市场，看看强劲如 Delphi 最终结果也是只能贱卖。那么，主宰 RIA 市场的关键因素是什么呢？我们也许无法预测。但正如上面所说，我们不能仅仅从技术方面来考虑，还需要更多的从市场来考虑，Delphi 就是前车之鉴，市场不认技术和经典，它是残酷无情的。

目前越来越多的软件产品走入开源模式，事实也证明开源对于软件产品的发展和开发者来说是双赢的。我们有理由相信随着 Silverlight 的开源，Silverlight 开发者和爱好者可以通过阅读和研究 Silverlight 的源代码，更深入的理解 Silverlight 并进一步增强它，从而利用 Silverlight 开发出更多更出色的 RIA 应用程序。因此，对于 Silverlight 来说，只有有更多 Web 开发者的加入才能让其拥有广泛的用户群体，才能加速 Silverlight 的普及。也就是说，开发社群活跃度将成为是否拥有更广泛用户群体的关键，也是一种技术能否生存的关键。

(2) 微软未来产品将立足于“网络化”

微软的 Windows 和 Office 在台式机应用程序开发领域赢得了极大的成功，但是在网络方面微软却没有什么优势可言。随着网络的高速发展，各种基于网络应用的市场前景也将越来越被看好。当微软的对手们在网络方面颇有建树后，微软肯定也不会对网络这块蛋糕犹豫不绝，它必定全力还击，甚至会进行更多的转型。而 Silverlight 在桌面和浏览器的结合中可以扮演一个很好的角色，它可以完美地把桌面程序“网络化”，这是未来的一个趋势。

RIA 技术 Silverlight 可以带给用户更丰富的“用户体验”，这是微软目前和未来产品所需要的。例如，通过 VB、C#、Python 等语言，微软可以让用户在 Silverlight 的框架上使用最新版本的 Office 办公软件、OneCare 杀毒软件等。也就是说，当未来所有的工作都趋向于网络化时，一切微软的程序都可以嵌入到 RIA 平台中，从而使到用户获得更好的界面效果、交互功能以及在线功能。因此，微软着力进行 RIA Silverlight 的开源宣传和推广，显然也是看到了富客户端技术的良好前景，而且 Silverlight 的开源模式也正好符合网络化模式的发展。

(3) 开源社区开发人员众多，更有创造力

现在有大量的开发人员加入到 Web 开发中来，而且这些 Web 开发人员大部分都活跃在全球的各种开源社区中。由于大家都在同样的圈子，关注同样的问题，于是自然而然的共享同样的代码，例如一些工具软件、插件、本地化项目等。这种模式在 Linux 操作系统身上已经被证明是非常有效的，当发现一个很好的创意时，很快就会得到共享和分享，Web 开发亦如此。当很多人为同一个目标努力工作时，那么这个目标是不是很快就会实现呢？简单的说，我们只要想象一下集体智慧的巨大力量，就能让我们为之激动。

总的来说，要想赢得 RIA 之战，就需要提供更多的技术和资源推广。在处于两强争霸的 RIA 战

役的转折点之时，只有拥抱开源社区的力量，才可以让微软拥有赢得 RIA 战争的动力。开源，不但是微软称霸 Web 网页开发的主动选择，也是顺应网络模式潮流的选择。至于开源后，微软未来之路如何走，我们不防拭目以待。

iPhone SDK4 惹怒开发者 老乔态度强硬

近日 iPhone SDK 4 的发布招致许多开发者的不满，不但有人在网络上笔谏，甚是有按耐不住，直接写信跟乔布斯沟通。其原因在于新的 iPhone SDK 4 Section 3.3.1 明确规定了 App 开发者不能使用（除了 C，C++，and Objective-C 以外的）第三方、跨平台程序工具、语言。



如果仍然打算给 iPhone，iPad 开发应用程序的话，那就别无选择。跨平台的语言如 Flash（使用 Adobe Flash CS5 语言）和 Mono Touch（使用微软.net 语言）是没戏了。另外，虽然 Qt 是 C++，但新的声明 only code written in C，C++，and Objective-C may compile and directly link against the Documented APIs 也明确封死了 Qt 移植到 iPhone 平台的可能性。

开发者只能以苹果规定的方式来使用文档中所提供的 API（应用编程接口），而无法使用或者调用私有 API。应用的原始代码必须使用 Objective-C、C、C++ 或 JavaScript 来编写才能够被 iPhone OS WebKit 引擎执行，而且只有用 C、C++ 以及 Objective-C 编写的代码才能够编译并直接与文档中已有的 API 建立连接。

Facebook 工程师乔伊休伊特（Joe Hewitt）称：这令我很烦恼。说实话，我觉得 Objective-C 很一般，我喜欢使用其他语言来开发有趣的 iPhone 应用。

一位名叫 Greg Slepak 的开发者，在跟乔布斯的通信中，不仅明确的表达了自己反对的立场，还引用了不少网友的批评文章。然而乔布斯的回应呢？乔布斯先是拿出一篇 Gruber（公认的苹果超级粉丝）写的新文章来反驳，里面 Gruber 算是替苹果为何设下如此限制来作个合理的推敲跟解释，同时也被老乔评为相当有见地。

Greg Slepak 随后则引 Firefox 的例子进行反驳，表示这样一个受大家欢迎、举足轻重的浏览器，也是允许其开发者使用各种跨平台的程序开发工具，乔布斯对此则是相当不以为然，乔布斯表示：过去我们也经历过类似的状况，不过从经验上来看，要是在软件开发者跟软件平台间，插入其它的中介开发工具，只会让编写出来的软件质量降低，并且限制了该平台的发展。

另外在 Greg Slepak 的博客上，他也针对 Gruber 的文章中提出的观点，做出了进一步的分析，结论还是站在应该保留第三方开发工具这个论点，一如 Mac 跟 Windows 上的一大票例子。

不过从老乔强硬的态度来看，这件事那怕吵得再凶，依然是无解，由于这一平台已经吸引了大量用户，因此无论开发者多么不情愿，也只能选择妥协。

Facebook 开放图谱 API 争议：被指非真正开放

4月26日，据美国社交媒体资讯网站 Mashable!报道，Facebook 最近推出的开放图谱 API (Open Graph API) 和贯穿互联网的“喜欢” (Like) 键引起了争议，有人争论这个平台并非真正“开放”。Facebook 就此表示，他们所作的是对互联网有利的。

该争论的演进过程如下：

1 开放图谱 (Open Graph)：Facebook 在上周三推出了开放图谱和“喜欢”键--用户可以在网络上任何一个地方点击“喜欢”，将这些数据分享到 Facebook 个人主页上。

2 对网站内容发布商/者的激励：发布商可将“喜欢”键添加到他们的网站上，还能添加其他 Facebook “社交化插件” (Social Plugins) (还有更多更复杂的开放图谱应用)。只要用户点击发布商网站页面上的“喜欢”键，就会在该用户的 Facebook 页面上显示一个链接。网站可以根据用户的朋友和他们“喜欢”的东西，来为该用户展示与之最相关的内容。

3 某些部分开放：发布商也在他们的网页上添加数据，来分辨特定的物品--比如分辨出一个信息是一首歌，还包括歌名和乐队名称 (语意数据)。这样能方便 Facebook 在互联网上轻松管理所有的数据--诸如 Google 之类的竞争对手也能获取这些语意数据。

4 某些部分封闭：对于用户点击了“喜欢”的数据，Facebook 的竞争者是得不到的--Google 和其他网站不能得到用户“喜欢”的网站的数据，除非在他们的网页上置入 Facebook 登陆口。因此，Facebook 在建立一个全世界人的喜好的数据，但是并不跟别人分享，除非别人在其网站上推广 Facebook (通过用 Facebook 的登陆口)。

5 其他的开放途径：如果不选择 Facebook 登陆口，还有其他的开放途径--比如 Open ID。但是网站发布商很少使用这些选择，因为他们并不能像 Facebook 登陆口一样带来流量和新的注册用户。

6 不能导出：Facebook 不允许用户一次性导出他们“喜欢”的东西。如果 Facebook 的竞争对手建立了一个更好的服务，用户想从 Facebook 上导出“喜欢”的东西，并导入到这个新的平台，用户是很难实现的。因此 Facebook 在建立一个关于用户的数据库，但用户并不能拥有它--Facebook 才拥有它。

7 选择更少：开放网络的支持者争论说，如果用户封锁在一个系统中，这对用户是不好的--竞争对手不能建立更好的系统和提供选择。

8 成功的关键：Facebook 的观点如果是正确的，前提是：除非 Facebook 将“社交图谱”封锁在自己的数据库中。这个公司 (Facebook) 不能用一个广受使用的身份 (ID) 系统来掌管整个互联网，并攫取价值 (比如挣很多钱)。

9 营销词汇：这暗示：既然开放图谱 API 并不是完全开放，Facebook 仅仅是将“开放”作为一个营销词汇。这从一定程度上说是对的，因为 Facebook 的价值命题是基于将数据“封锁”。

10 Facebook 赢了：Facebook 已经通过为用户和发布商解决身份 (ID) 问题而赢得了整个网络。他们同时促进了语意网络的发展。竞争对手会用一个真正开放的方式来反击，但是不会提供适当的激励措施 (为发布商带来流量)，那他们如何能竞争？

Linux 基金会董事：未来 2-5 年硬件将会免费

据国外媒体报道，在加州周三召开的 Linux 基金会年度合作峰会上，基金会执行董事吉姆·泽姆

林（Jim Zemlin）以独特的视角对当今 Linux 市场进行了阐述，并预言 2-5 年里硬件将会变成免费的。

泽姆林论述的主旨是：Linux 正越来越强大。他表示，宏观经济状况有利于 Linux 和开源软件的发展。人们普遍认为，Linux 的获得和运行比专有软件更加便宜，这不仅适用于终端用户，而且适用于设备制造商和软件开发系统。



如果当初谷歌使用微软的 .NET 等专有技术，它能够成为今天的谷歌吗？很可能不会。任何形式的服务商（如云计算提供商）都需要保持对平台的控制，而只有 Linux 和开源软件才允许这种控制权。

由于市场上出现的新设备，同时由于客户端计算的性质正发生变化，Linux 的吸引力正在增加——特别是对移动互联网来说。问题又回到控制权方面，通过使用 Linux，设备制造商在应用、服务和经济方面将拥有更大的控制权。

如果企业有自己的软件，就可以用更新颖的方式拓展市场——电子阅读器、智能手机等就利用了一些新的经济现象。

泽姆林表示，新 PC 经济更类似于手机产业，它相对于传统的 PC 价值链已经发生了变化。其中的一个例子是，苹果将从 App Store 商店的毛收入中获得 30% 的分成。新的模式将价值从平台本身转移到了其中的应用上了。对 Linux 来说，另一个重大变化是服务商正开始提供额外的、更具吸引力的定制化服务。

泽姆林预计，在接下来的 2-5 年里硬件将会变成免费的。人们将不再购买软件或硬件，而是会购买服务，比如亚马逊的在线服务。对硬件提供商来说这并非坏事，因为随着越来越多的服务以在线的方式提供，他们将可以把设备卖给新的客户。

总之，Linux 生态系统的发展已经超出了过去十年里人们的预料。而从 Linux 合作峰会看来，IT 业界向 Linux 和开源软件的转移才刚刚开始。

Java 之父黯然离职 开源将何去何从

Java 创始人詹姆斯·高斯林 (James Gosling) 日前在博客中称, 已经在 4 月 2 日从甲骨文 (Oracle) 退休。很多人的第一反应是, 他是否会投奔微软、Google 之类的公司呢? 不过 Gosling 表示目前还没有进一步打算。



关于离职的具体原因, 高斯林没有透露。只是在博客中写道: “是的, 那些传言都是真的: 我已经在上周 (4 月 2 日) 从 Oracle 辞职。我要对周三参加圣彼得堡 TechDays 希望听我演讲的所有人道歉。没能出席, 我感到非常难受。至于离开的原因, 我很难回答: 能说的任何准确而且真实的话都弊大于利。最困难的地方, 是无法再与这些年来我有幸一起工作的所有人共事了。除了在找工作之前休息一段时间之外, 我不知道下一步会做些什么。”

高斯林辞职前担任甲骨文客户端软件集团的首席技术官, 而在加盟甲骨文之前, 高斯林担任 Sun 公司产品开发部首席技术官。此不到一个月前, 高斯林还强调了 Java 对于甲骨文的重要性。高斯林在拉斯维加斯的一次 Java 研讨会上说: “甲骨文肯定会尽全力保持 Java 及整个 Java 系统的活力并使其持续健康发展。”而谈到个人时, 他当时说希望自己 2030 年的时候仍在写代码。而今年 1 月已开始有传言说 Gosling 将离开。

今年 1 月份甲骨文收购 Sun 的交易完成后, 多位前 Sun 高管已经从甲骨文离职, 其中包括前 Sun 首席执行官乔纳森·施瓦茨 (Jonathan Schwartz) 以及 XML 发明人蒂姆·布雷 (Tim Bray), 高斯林是最近从甲骨文离职的一名前 Sun 高管。

高斯林的离开让业界对于甲骨文对待开源技术的诚意再次产生了怀疑。

从开源到收费: 甲骨文榨取 Solaris 最后价值

09 年 4 月 20 日晚, 甲骨文宣布以每股 9.5 美元的价格收购 Sun。该交易价值约 74 亿美元。还差 11 天, 甲骨文收购 Sun 满一年, 让我们看看甲骨文对 sun 做了些什么。

MySQL 难撼动 承诺终成空?

早在收购之处, 众多评论家就将 MySQL 看做是甲骨文收购 Sun 的一个重要原因, 而以埃里森的个性而言, 威胁的存在为什么不融合呢? 他一向喜欢融合被收购公司的产品和技术, 此前就对

People Soft 的收购就是一个比较典型的例子，尽管甲骨文保留了 PeopleSoft（仁科）的商标，但对这家公司知之甚少的企业很难再感受到仁科存在。这也造成了前期欧盟迟迟不通过甲骨文收购 Sun 交易审查的主要原因。

但是这次甲骨文却不得不做出一些让步，2009 年 12 月，甲骨文做出了 10 项具体承诺以此来支持 MySQL 的后续发展。甲骨文表示，未来 3 年内至少投资 7200 万美元用于开发 MySQL，以及继续发布这款开源软件的最新技术研发成果。在未来 5 年内允许其它技术厂商继续在各自产品中授权使用 MySQL。显然 MySQL 强大的用户基础让甲骨文还不能轻言融合。

OpenOffice 阵营面临瓦解！

甲骨文公司首席社区架构师 Edward Screven 曾表示，OpenOffice.org 将作为一个独立的业务部门来管理，Sun 的开发和支持团队会继续保留。甲骨文将继续支持免费的 OpenOffice.org 社区。但是甲骨文还计划推出名为 Oracle 云办公的软件套装，甲骨文已经开发这个软件有一段时间了。但是原来大量借鉴 OpenOffice.org 的 StarOffice 却没有被提及，这是原来用于和 IBM 竞争的一款产品。但是显然这并不是一个好说辞

Solaris 的最后剩余价值

甲骨文收购 Sun 之后对 Solaris 鲜有动作，尽管 OpenSolaris 在去年按时更新，但显然今年的更新可能会杳无音讯，因为 Solaris 将不再免费，不购买商业支持合同就不能使用 Solaris，这将使那些 Solaris 的大客户们重新考虑其他的选择。Sun 曾经努力地尝试减缓甚至扭转 Linux 蚕食 Unix 的趋势，而甲骨文现在正用一句文字来抹消 Sun 之前所有的努力，不仅如此，甲骨文这样的举动将成功转移 Solaris 的用户到自己的 Unbreakable Linux 上，这无疑是榨取 Solaris 最后的剩余价值。

依稀记得甲骨文收购 Sun 的时候，甲骨文 CEO 拉里·埃里森（Larry Ellison）说，“我们收购 Sun 将改变 IT 业，整合第一流的企业软件和关键任务计算系统。甲骨文将成为业界唯一一家提供综合系统的厂商，系统的性能、可靠性和安全性将有所提高，而价格将会下滑。”

现在回味一下，寓意很深刻，整合的意义：弱化、肢解、融合、消失。

大型机 Linux：风雨十周年

2010 年是 Linux 用于大型机的第 10 个年头，本文将为大家介绍 Linux 进入大型机 10 年来的历史，它支持的第一个应用程序，优点，以及目前的市场前景，拥有成本和可用的应用，另外还对大型机 Linux 是否适合你数据中心的虚拟化项目提供了很好的建议。

1999 年是大型机 Linux 的元年，IBM 和 SUSE（2004 年被 Novell 收购）达成一项协议，宣布共同推出第一个用于大型机的 Linux。到 2000 年，第一个企业级版本就绪：用于 S/390 的 SUSE Linux Enterprise Server。它的第一个重要的客户是来自北欧的电信公司 Telia。2010 年是 Linux 用于大型机的 10 周年，Linux 用于大型机的价值主张从 2000 年到现在一样被看得很重要。

大型机 Linux 最开始是两个独立的移植 Linux 到 IBM 大型机的项目。第一个是 Bigfoot (i370) 移植项目，于 1998 年 8 月由 Linus Vepstas 启动，Vepstas 和他的同事使用 IBM/370 大型机在普林斯顿进行移植，但因政治、社会和市场原因，Bigfoot 项目被停止了。之后 IBM 又宣布了第二个 Linux 向大型机移植项目，即 Linux 向 S/390 移植。向 S/390 移植 Linux 始于 1998 年，它是 IBM 在德国的 Boeblingen 实验室的一个失败项目，但 IBM 一直未对外公开这个失败的项目，保密期长达一年多。人们猜测当初保密可能是开发人员害怕来自 IBM 其它部门的报复，因此 S/390 团队未与 Bigfoot 团队一起工作。S/390 项目直到 1999 年 12 月 18 日才公开，它基于 Linux 2.2.13 内核。

第一款可商业化用于 S/390 的 Linux 来自德国的 SUSE，1999 年 SUSE 开始和 IBM 合作，主要工作是在德国的 IBM Boeblingen 实验室和 Marist 大学完成的，Boeblingen 实验室到 SUSE 总部 Nuremberg 开车只有两小时的路程，这就是为什么在 S/390 移植项目上 IBM 选择 SUSE 的原因。在向 S/390 移植 Linux 的时间里，SUSE 已经在内部开始了很多不同的 Linux 移植项目，包括向 powerPC，x86，Alpha 和 SPARC 移植。

1999 到 2000 年，当 SUSE 向 IBM 大型机移植 Linux 时，未让其它大型机厂商参与。向大型机移植 Linux 包含创建新的过程和基础设施，新的商业模式，24/7 全球支持，同步 IBM/SUSE 三级支持过程，以及 ISV 和 IHV 认证等等。当时富士通和日立对 x86 和 IA-64（安腾）平台更感兴趣，即使他们的大型机与 IBM 的大型机兼容。



为 S/390 构建的 SUSE Linux Enterprise Server 早期版本可运行在 Amdahl 和 Comparex 大型机上，但在 2000 年时大部分大型机用户都使用的是 IBM 大型机，今天也如此。为了让 SUSE Linux Enterprise Server 运行在 S/390 大型机上，Marcus Kraft（当时的 SUSE 开发经理）说 SUSE 需要一个编译器，库和一些基础包，SUSE 的工程团队从 Marist 大学取得了大部分需要的资源，在 Marist 大学的帮助下，SUSE Linux Enterprise Server 终于顺利运行在了 S/390 上。

接下来是继续开发和测试，使之成为一个真正可行的商业化产品，SUSE 从 IBM 获得了一台大型机，在 SUSE 自动构建（Autobuild）系统的帮助下，大约一周时间就可以调试好在一台大型机运行 Linux。

由于在 S/390 的 Linux 驱动知识产权方面可能存在法律问题，最后 IBM 不得不亲自编写驱动代码，自那以后知识产权问题就被解决了，现在所有代码都根据 GPL 协议公开了。为了在正式发布适用于 S/390 的 SUSE Linux Enterprise Server 之前找到对大型机 Linux 感兴趣的客户，IBM 和 SUSE 举行了一个长达两周的安装活动，因为参与者无法携带大型机，因此通过电话会议的形式讲述了安装步骤，当初参加这个活动的客户至今仍然有很多还是 SUSE 的忠实用户。

本期推荐

谁是 MySQL 的新主人

-----2010 MySQL 全球用户大会观感

谁是 MySQL 的新主人？这似乎是个伪命题，MySQL 的新主人无疑是那个强势的甲骨文公司。去年对 Sun 的收购，让甲骨文顺利的将一个潜在的数据库对手收入囊中，开源社区大哗，一时众说纷纭，唱衰者有之、看好者也有之，但总体看来，忧虑的情绪在社区中弥散，数位 MySQL 创始人的“拯救”行动，更将 MySQL 的悲壮推向了高潮。

从 05 年起，笔者就从开源观察者的角度转变成一个正式的开源从业者，从事的，正是 MySQL 的商业业务，经历了中国 MySQL 从民间到官方、从互联网向传统行业进军的过程。5 年间 MySQL 数次起落，作为”坐过山车的人”，比起旁观者，可能更多一层体会。在国内等 Oracle 的消息，感觉是那么的不可靠，为了了解真相，了解 MySQL 发展的趋势，2010 年 4 月 11 日，我们来到加州小城 Santa Clara，参加 2010 年 MySQL 全球用户大会。

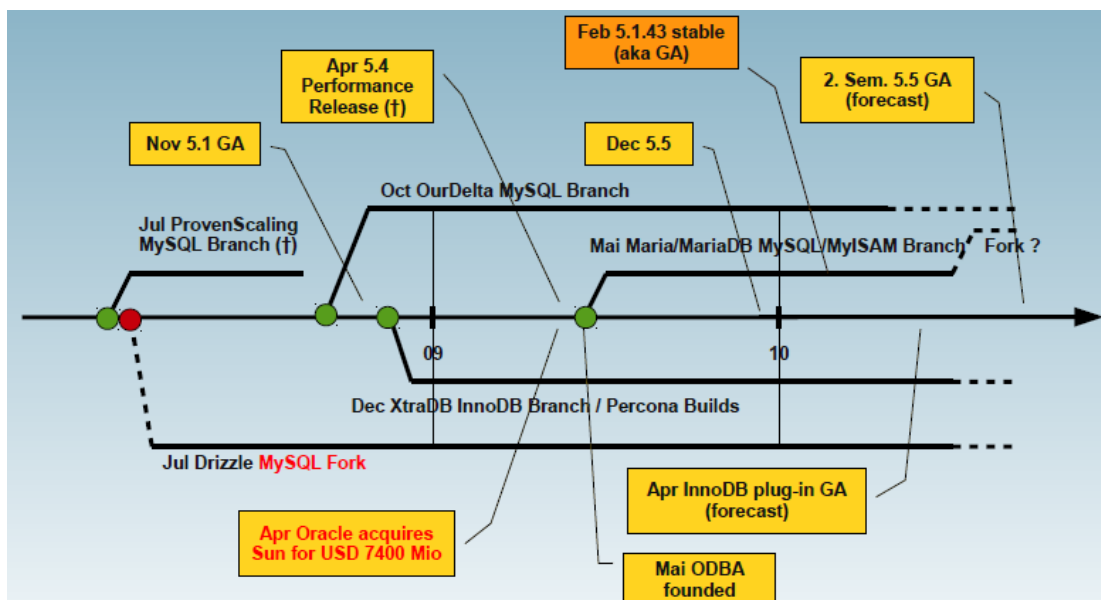


这次包括 Percona, Inforbright, Zmanda, Pentaho 等 MySQL 关联厂商、技术分支以及甲骨文全球的技术参加了大会，颇有“华山论剑”的味道，可惜国内仅有我们一行两人，遇上了原先 MySQL 亚太公司的两位“领导”，原来他们也随着 Sun 进入 Oracle 的体系了，他们将在今年 5 月来中国继续他们的 MySQL 事业，这点颇令我们惊喜。



大名鼎鼎的 MySQL Performance Blog 也在大会摆摊

MySQL 用户大会是 MySQL 开源社区的活动，美国 MySQL 社区“情绪稳定”，由于 MySQL 开源属性不变，包括 Percona 在内的厂商都希望从 MySQL 的变化中占据更多先机，在今年 MySQL 用户大会上上演了精彩一幕。MySQL Performance Blog 充分展示了 MySQL 技术服务咨询能力，从服务器硬件选型、数据库调优及解决方案都形成了自己的服务体系，特别在性能方面，XtraDB 全面超越自带的 InnoDB 及 InnoDB Plugin，越来越多的用户选择专业的 MySQL 技术支持服务团队作为公司数据库技术的后备保障。同时，Oracle 在 MySQL 用户大会期间，在 MySQL 5.5-m3 发布没几天的情况下，紧急发布了 MySQL 5.5-m4，并强调了新版性能上的调整及 InnoDB Plugin 新特性。甲骨文的首席架构师 Edward Screven 更在大会上表示 MySQL 对于甲骨文来说是十分重要的一部分，表达对 MySQL 的关切和支持，并明确表示 Oracle 计划增加对 MySQL 的投资，以进一步完善 MySQL 的功能，改善 MySQL 的性能。



从上图可以看出，图中中轴线是 MySQL 官方版本今年的发展路线，其中 Google MySQL team 在 MySQL 5.0.37 的基础上发布了大量的补丁（主要针对 InnoDB），使得 MySQL 的健壮性和性能都有了极大的提高。而其他 MySQL 分支也在持续发展，对 MySQL 官方版本的发展起到推动和鞭策作用，在开源社区发展出的力量和 Oracle 体系的竞争中，MySQL 本身呈现出加速发展的趋势，无论

MySQL 的主人是谁，MySQL 的开源属性都无法改变，Oracle 体系下 MySQL 变的更完善、更稳定、更加商业化，社区的分支版本更加开放、具备更强的关键性能，也许这两条并行的发展路径，能够创生出更加强大的新 MySQL。

无论是 Oracle 这样的商业大亨，还是机动灵活的开源社区，只要大家共同推动 MySQL 的技术进步，对我们广大 MySQL 的爱好者和用户来说都是好消息，我们不关心谁是 MySQL 的新主人，我们会持续不断的跟踪最新的 MySQL 新技术和新发展，并把专业化的技术支持和产品服务带给国内的广大爱好者和用户，共同分享新技术成果带来的新体验和新快乐！

技术新知

Linux 驱动开发方法论

fudan_abc 之 任桥伟

有一种感动，叫内牛满面，有一种机制，叫模块机制。显然，这种模块机制给那些 Linux 的发烧友们带来了方便，因为模块机制意味着人们可以把庞大的 Linux 内核划分为许许多多小的模块。对于编写设备驱动程序的开发来说，从此以后他们可以编写设备驱动程序却不需要把她编译进内核，不用 reboot 机器，她只是一个模块，当你需要她的时候，你可以把她抱入怀中（insmod），当你不再需要她的时候，你可以把她一脚踢开（rmmod）。

于是，忽如一夜春风来，内核处处是模块。让我们从一个伟大的例子去认识模块。这就是传说中的"Hello World!"，这个梦幻般的名字我们看过无数次了，每一次她出现在眼前，就意味着我们开始接触一种新的计算机语言了。（某程序员对书法十分感兴趣，退休后决定在这方面有所建树。于是花重金购买了上等的文房四宝。一日，饭后突生雅兴，一番磨墨拟纸，并点上了上好的檀香，颇有王羲之风范，又具颜真卿气势，定神片刻，泼墨挥毫，郑重地写下一行字：hello world）

请看下面这段代码，她就是 Linux 下的一个最简单的模块。当你安装这个模块的时候，她会用她特有的语言向你表白：“Hello, world! ”，而后来你卸载了这个模块，你无情抛弃了她，她很伤心，她很绝望，但她没有抱怨，她只是淡淡地说，“Goodbye, cruel world! ”（再见，残酷的世界!）

```
/****** hello.c *****/

1 #include <linux/init.h> /* Needed for the macros */
2 #include <linux/module.h> /* Needed for all modules */
3 MODULE_LICENSE("Dual BSD/GPL");
4 MODULE_AUTHOR("fudan_abc");
5
6 static int __init hello_init(void)
7 {
8     printk(KERN_ALERT "Hello, world!\n");
9     return 0;
10 }
11
12 static void __exit hello_exit(void)
13 {
14     printk(KERN_ALERT "Goodbye, cruel world\n");
15 }
16
17 module_init(hello_init);
18 module_exit(hello_exit);
```

你需要使用 module_init() 和 module_exit()，你可以称它们为函数，不过实际上它们是一些宏，你可以不用去知道她们背后的故事，只需要知道，在 Linux Kernel 2.6 的世界里，你写的任何一个模块

都需要使用它们来初始化或退出，或者说注册以及后来的注销。

当你用 `module_init()` 为一个模块注册了之后，在你使用 `insmod` 这个命令去安装的时候，`module_init()` 注册的函数将会被执行。而当你用 `rmmod` 这个命令去卸载一个模块的时候，`module_exit()` 注册的函数将会被执行。`module_init()` 被称为驱动程序的初始化入口 (driver initialization entry point)。

怎么样演示以上代码的运行呢？没错，你需要一个 Makefile。

```
1 # To build modules outside of the kernel tree, we run "make"
2 # in the kernel source tree; the Makefile these then includes this
3 # Makefile once again.
4 # This conditional selects whether we are being included from the
5 # kernel Makefile or not.
6 ifeq ($(KERNELRELEASE),)
7
8   # Assume the source tree is where the running kernel was built
9   # You should set KERNELDIR in the environment if it's elsewhere
10  KERNELDIR ?= /lib/modules/$(shell uname -r)/build
11  # The current directory is passed to sub-makes as argument
12  PWD := $(shell pwd)
13
14 modules:
15     $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
16
17 modules_install:
18     $(MAKE) -C $(KERNELDIR) M=$(PWD) modules_install
19
20 clean:
21     rm -rf *.o *~ core .depend *.cmd *.ko *.mod.c .tmp_versions
22
23 .PHONY: modules modules_install clean
24
25 else
26   # called from kernel build system: just declare what our modules are
27   obj-m := hello.o
28 endif
```

在 lwn.net 上可以找到这个例子，你可以把以上两个文件放在你的某个目录下，然后执行 `make`，也许你不一定能成功，因为 Linux Kernel 2.6 要求你编译模块之前，必须先在内核源代码目录下执行 `make`，换言之，你必须先配置过内核，执行过 `make`，然后才能 `make` 你自己的模块。原因就不细说了，你按着要求的这么去做就行了。在内核顶层目录 `make` 过之后，你就可以在你当前放置 Makefile 的目录下执行 `make` 了。`make` 之后你就应该看到一个叫做 `hello.ko` 的文件生成了，恭喜你，这就是你将要测试的模块。

执行命令，

```
#insmod hello.ko
```

同时在另一个窗口，用命令 `tail -f /var/log/messages` 察看日志文件，你会看到 Hello world 被打印了出来。再执行命令，

```
#rmmod hello.ko
```

此时，在另一窗口你会看到 Goodbye, cruel world! 被打印了出来。

到这里，我该恭喜你，因为你已经能够编写 Linux 内核模块了。这种感觉很美妙，不是吗？你可以嘲笑秦皇汉武略输文采唐宗宋祖稍逊风骚，还可以嘲笑一代天骄成吉思汗只识弯弓射大雕了。是的，阿娇姐姐告诉我们，只要我喜欢，还有什么不可以。

日后我们会看到，2.6 内核中，每个模块都是以 `module_init` 开始，以 `module_exit` 结束。对大多数人来说没有必要知道这是为什么，记住 就可以了，对大多数人来说，这就像是 1+1 为什么等于 2 一样，就像是两点之间最短的是直线，不需要证明，如果一定要证明两点之间直线最短，可以扔一块骨头 在 B 点，让一条狗从 A 点出发，你会发现狗走的是直线，是的，狗都知道，咱还能不知道吗？

对于驱动开发来说，设备模型的理解是根本，毫不夸张得说，理解了设备模型，再去那些五花八门的驱动程序，你会发现自己站在了另一个高度，从而有了一种俯视的感觉，就像凤姐俯视知音和故事会，韩峰同志俯视女下属。

顾名思义就知道设备模型是关于设备的模型，既不是任小强们的房模，也不是张导的炮模。对咱们写驱动的和不用写驱动的人来说，设备的概念就是总线及其相连的各种设备了。电脑城的 IT 工作者都会知道设备是通过总线连到计算机上的，而且还需要对应的驱动才能用，可是总线是如何发现设备的，设备又是如何和 驱动对应起来的，它们经过怎样的艰辛才找到命里注定的那个他，它们的关系如何，白头偕老型的还是朝三暮四型的，这些问题就不是他们关心的了，而是咱们需要 关心的。在房市股市千锤百炼的咱们还能够惊喜的发现，这些疑问的中心思想中心词汇就是总线、设备和驱动，没错，它们就是咱们这里要聊的 Linux 设备模型 的名角。

总线、设备、驱动，也就是 bus、device、driver，既然是名角，在内核里都会有它们自己专属的结构，在 `include/linux/device.h` 里定义。

```
52 struct bus_type {
53     const char      * name;
54     struct module    * owner;
55
56     struct kset      subsys;
57     struct kset      drivers;
58     struct kset      devices;
59     struct klist      klist_devices;
60     struct klist      klist_drivers;
61
62     struct blocking_notifier_head bus_notifier;
63
64     struct bus_attribute * bus_attrs;
65     struct device_attribute * dev_attrs;
66     struct driver_attribute * drv_attrs;
```



```
67 struct bus_attribute drivers_autoprobe_attr;
68 struct bus_attribute drivers_probe_attr;
69
70 int (*match)(struct device * dev, struct device_driver * drv);
71 int (*uevent)(struct device *dev, char **envp,
72 int num_envp, char *buffer, int buffer_size);
73 int (*probe)(struct device * dev);
74 int (*remove)(struct device * dev);
75 void (*shutdown)(struct device * dev);
76
77 int (*suspend)(struct device * dev, pm_message_t state);
78 int (*suspend_late)(struct device * dev, pm_message_t state);
79 int (*resume_early)(struct device * dev);
80 int (*resume)(struct device * dev);
81
82 unsigned int drivers_autoprobe:1;
83 };
124 struct device_driver {
125     const char * name;
126     struct bus_type * bus;
127
128     struct kobject kobj;
129     struct klist klist_devices;
130     struct klist_node knode_bus;
131
132     struct module * owner;
133     const char * mod_name; /* used for built-in modules */
134     struct module_kobject * mkobj;
135
136     int (*probe) (struct device * dev);
137     int (*remove) (struct device * dev);
138     void (*shutdown) (struct device * dev);
139     int (*suspend) (struct device * dev, pm_message_t state);
140     int (*resume) (struct device * dev);
141 };
407 struct device {
408     struct klist klist_children;
409     struct klist_node knode_parent; /* node in sibling list */
410     struct klist_node knode_driver;
411     struct klist_node knode_bus;
412     struct device *parent;
413
```

```
414 struct kobject kobj;
415 char bus_id[BUS_ID_SIZE]; /* position on parent bus */
416 struct device_type *type;
417 unsigned is_registered:1;
418 unsigned uevent_suppress:1;
419
420 struct semaphore sem; /* semaphore to synchronize calls to
421    * its driver.
422    */
423
424 struct bus_type * bus; /* type of bus device is on */
425 struct device_driver *driver; /* which driver has allocated this
426    device */
427 void *driver_data; /* data private to the driver */
428 void *platform_data; /* Platform specific data, device
429    core doesn't touch it */
430 struct dev_pm_info power;
431
432 #ifdef CONFIG_NUMA
433 int numa_node; /* NUMA node this device is close to */
434 #endif
435 u64 *dma_mask; /* dma mask (if dma'able device) */
436 u64 coherent_dma_mask; /* Like dma_mask, but for
437    alloc_coherent mappings as
438    not all hardware supports
439    64 bit addresses for consistent
440    allocations such descriptors. */
441
442 struct list_head dma_pools; /* dma pools (if dma'ble) */
443
444 struct dma_coherent_mem *dma_mem; /* internal for coherent mem
445    override */
446 /* arch specific additions */
447 struct dev_archdata archdata;
448
449 spinlock_t devres_lock;
450 struct list_head devres_head;
451
452 /* class_device migration path */
453 struct list_head node;
454 struct class *class;
455 dev_t devt; /* dev_t, creates the sysfs "dev" */
456 struct attribute_group **groups; /* optional groups */
```

457

```
458 void (*release)(struct device * dev);  
459 };
```

有没有发现它们的共性是什么？对，不是很傻很天真，而是很长很复杂。不过不妨把它们看成艺术品，既然是艺术，当然不会让你那么容易的就看懂了，不然怎么称大师称名家。这么想想咱们就会比较的宽慰了，阿 Q 是鲁迅对咱们 80 后最大的贡献。

我知道进入了 21 世纪，最缺的就是耐性，房价股价都让咱们没有耐性，内核的代码也让人没有耐性。不过做为最没有耐性的一代人，还是要平心静气的扫一下上面的结构，我们会发现，struct bus_type 中有成员 struct kset drivers 和 struct kset devices，同时 struct device 中有两个成员 struct bus_type * bus 和 struct device_driver * driver，struct device_driver 中有两个成员 struct bus_type * bus 和 struct klist klist_devices。先不说什么是 klist、kset，光从成员的名字看，它们就是一个完美的三角关系。我们每个人心中是不是都有两个她？一个梦中的她，一个现实中的她。

凭一个男人的直觉，我们可以知道，struct device 中的 bus 表示这个设备连到哪个总线上，driver 表示这个设备的驱动是什么，struct device_driver 中的 bus 表示这个驱动属于哪个总线，klist_devices 表示这个驱动都支持哪些设备，因为这里 device 是复数，又是 list，更因为一个驱动可以支持多个设备，而一个设备只能绑定一个驱动。当然，struct bus_type 中的 drivers 和 devices 分别表示了这个总线拥有哪些设备和哪些驱动。

单凭直觉，张钰红不了。我们还需要看看什么是 klist、kset。还有上面 device 和 driver 结构里出现的 kobject 结构是什么？作为一个五星红旗下长大的孩子，我可以肯定的告诉你，kobject 和 kset 都是 Linux 设备模型中最基本的元素，总线、设备、驱动是西瓜，kobject、klist 是种瓜的人，没有幕后种瓜人的汗水不会有清爽解渴的西瓜，我们不能光知道西瓜的甜，还要知道种瓜人的辛苦。kobject 和 kset 不会在意自己的得失，它们存在的意义在于把总线、设备和驱动这样的对象连接到设备模型上。种瓜的人也不会在意自己的汗水，在意的只是能不能送出甜蜜的西瓜。

一般来说应该这么理解，整个 Linux 的设备模型是一个 OO 的体系结构，总线、设备和驱动都是其中鲜活存在的对象，kobject 是它们的基类，所实现的只是一些公共的接口，kset 是同种类型 kobject 对象的集合，也可以说是对象的容器。只是因为 C 里不可能会有 C++ 里类的 class 继承、组合等的概念，只有通过 kobject 嵌入到对象结构里来实现。这样，内核使用 kobject 将各个对象连接起来组成了一个分层的结构体系，就好像马列主义 将我们 13 亿人也连接成了一个分层的社会体系一样。kobject 结构里包含了 parent 成员，指向了另一个 kobject 结构，也就是这个分层结构的上一层结点。而 kset 是通过链表来实现的，这样就可以明白，struct bus_type 结构中的成员 drivers 和 devices 表示了一条总线拥有两条链表，一条是设备链表，一条是驱动链表。我们知道了总线对应的数据结构，就可以找到这条总线关联了多少设备，又有哪些驱动来支持这类设备。

那么 klist 呢？其实它就包含了一个链表和一个自旋锁，我们暂且把它看成链表也无妨，本来在 2.6.11 内核里，struct device_driver 结构的 devices 成员就是一个链表类型。这么一说，咱们上面的直觉都是正确的，如果买股票，摸彩票时直觉都这么管用，就不会有咱们这被压扁的一代了。

现在的人都知道，三角关系很难处。那么总线、设备和驱动之间是如何和谐共处那？先说说总线中的那两条链表是怎么形成的。内核要求每次出现一个设备就要向总线汇报，或者说注册，每次出现一个驱动，也要向总线汇报，或者说注册。比如系统初始化的时候，会扫描连接了哪些设备，并为每一个设备建立起一个 struct device 的变量，每一次有一个驱动程序，就要准备一个 struct device_driver 结构的变量。把这些变量统统加入相应的链表，device 插入 devices 链表，driver

插入 drivers 链表。这样通过总线就能找到每一个设备，每一个驱动。然而，假如计算机里只有设备却没有对应的驱动，那么设备无法工作。反过来，倘若只有驱动却没有设备，驱动也起不了任何作用。在他们遇见彼此之前，双方都如同路埂的野草，一个飘啊飘，一个摇啊摇，谁也不知道未来在哪里，只能在生命的风里飘摇。于是总线上的两张表里就慢慢的就挂上了那许多孤单的灵魂。devices 开始多了，drivers 开始多了，他们像是来自两个世界，devices 们彼此取暖，drivers 们一起狂欢，但他们有一点是相同的，都只是在等待属于自己的那个另一半。

现在，总线上的两条链表已经有了，这个三角关系三个边已经有了两个，剩下的那个那？链表里的设备和驱动又是如何联系那？先有设备还是先有驱动？很久很久以前，在那激情燃烧的岁月里，先有的是设备，每一个要用的设备在计算机启动之前就已经插好了，插放在它应该在的位置上，然后计算机启动，然后操作系统开始初始化，总线开始扫描设备，每找到一个设备，就为其申请一个 struct device 结构，并且挂入总线中的 devices 链表中来，然后每一个驱动程序开始初始化，开始注册其 struct device_driver 结构，然后它去总线的 devices 链表中去寻找(遍历)，去寻找每一个还没有绑定驱动的设备，即 struct device 中的 struct device_driver 指针仍为空的设备，然后它会去观察这种设备的特征，看是否是他所支持的设备，如果是，那么调用一个叫做 device_bind_driver 的函数，然后他们就结为了秦晋之好。换句话说，把 struct device 中的 struct device_driver driver 指向这个驱动，而 struct device_driver driver 把 struct device 加入他的那张 struct klist klist_devices 链表中来。就这样，bus、device 和 driver，这三者之间或者说他们中的两两之间，就给联系上了。知道其中之一，就能找到另外两个。一荣俱荣，一损俱损。

但现在情况变了，在这红莲绽放的日子里，在这樱花伤逝的日子里，出现了一种新的名词，叫热插拔。设备可以在计算机启动以后在插入或者拔出计算机了。因此，很难再说是先有设备还是先有驱动了。因为都有可能。设备可以在任何时刻出现，而驱动也可以在任何时刻被加载，所以，出现的情况就是，每当一个 struct device 诞生，它就会去 bus 的 drivers 链表中寻找自己的另一半，反之，每当一个 struct device_driver 诞生，它就去 bus 的 devices 链表中寻找它的那些设备。如果找到了合适的，那么 OK，和之前那种情况一下，调用 device_bind_driver 绑定好。如果找不到，没有关系，等待吧，等到昙花再开，等到风景看透，心中相信，这世界上总有一个人是你所等的，只是还没有遇到而已。

设备模型拍得再玄幻，它也只是个模型，必须得落实在具体的子系统，否则就只能抱着个最佳技术奖空遗憾。既然前面已经以 USB 子系统的实现分析示例了分析内核源码应该如何入手，那么这里就仍然以 USB 子系统为例，看看设备模型是如何软着陆的。

内核中 USB 子系统的结构

我们已经知道了 USB 子系统的代码都位于 drivers/usb 目录下面，也认识了一个很重要的目录——core 子目录。现在，我们再来看一个很重要的模块——usbcore。你可以使用“lsmod”命令看一下，在显示的结果里能够找到有一个模块叫做 usbcore。

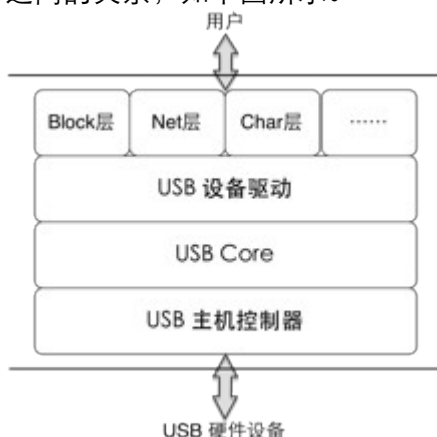
```
localhost:/usr/src/linux-2.6.23/drivers/usb/core # lsmod
```

Module	Size	Used by
af_packet	55820	2
raw	89504	0
nfs	230840	2
lockd	87536	2 nfs
nfs_acl	20352	1 nfs
sunrpc	172360	4 nfs,lockd,nfs_acl

```
ipv6          329728 36
button        24224 0
battery       27272 0
ac            22152 0
apparmor      73760 0
aamatch_pcre  30720 1 apparmor
loop          32784 0
usbhid        60832 0
dm_mod        77232 0
ide_cd        57120 0
hw_random     22440 0
ehci_hcd      47624 0
cdrom         52392 1 ide_cd
uhci_hcd      48544 0
shpchp       61984 0
bnx2          157296 0
usbcore       149288 4 usbhid,ehci_hcd,uhci_hcd
e1000         130872 0
pci_hotplug   44800 1 shpchp
reiserfs      239616 2
edd           26760 0
fan           21896 0
.....
```

找到了 usbcore 那一行吗？core 就是核心，基本上你要在你的电脑里用 USB 设备，那么两个模块是必须的：一个是 usbcore，这就是核心 模块；另一个是主机控制器的驱动程序，比如这里 usbcore 那一行我们看到的 ehci_hcd 和 uhci_hcd，你的 USB 设备要工作，合适的 USB 主机控制器模块也是必不可少的。

usbcore 负责实现一些核心的功能，为别的设备驱动程序提供服务，提供一个用于访问和控制 USB 硬件的接口，而不用去考虑系统当前存在哪种主机控制器。至于 core、主机控制器和 USB 驱动三者之间的关系，如下图所示。



USB 驱动和主机控制器就像 core 的两个保镖，协议里也说了，主机控制器的驱动（HCD）必须位

于 USB 软件的最下一层。HCD 提供主机控制器硬件的抽象，隐藏硬件的细节，在主机控制器之下是物理的 USB 及所有与之连接的 USB 设备。而 HCD 只有一个客户，对一个人负责，就是 usbcore。usbcore 将用户的请求映射到相关的 HCD，用户不能直接访问 HCD。

core 为咱们完成了大部分的工作，因此咱们写 USB 驱动的时候，只能调用 core 的接口，core 会将咱们的请求发送给相应的 HCD。

USB 子系统与设备模型

关于设备模型，最主要的问题就是，bus、device、driver 是如何建立联系的？换言之，这三个数据结构中的指针是如何被赋值的？绝对不可能发生的事情是，一旦为一条总线申请了一个 struct bus_type 的数据结构之后，它就知道它的 devices 链表和 drivers 链表会包含哪些东西，这些东西一定不会是先天就有的，只能是后天填进来的。

具体到 USB 子系统，完成这个工作的就是 USB core。USB core 的代码会进行整个 USB 系统的初始化，比如申请 struct bus_type usb_bus_type，然后会扫描 USB 总线，看线上连接了哪些 USB 设备，或者说 Root Hub 上连了哪些 USB 设备，比如说连了一个 USB 键盘，那么就为它准备一个 struct device，根据它的实际情况，为这个 struct device 赋值，并插入 devices 链表中来。

又比如 Root Hub 上连了一个普通的 Hub，那么除了要为这个 Hub 本身准备一个 struct device 以外，还得继续扫描看这个 Hub 上是否又连了别的设备，有的话继续重复之前的事情，这样一直进行下去，直到完成整个扫描，最终就把 usb_bus_type 中的 devices 链表给建立了起来。

那么 drivers 链表呢？这个就不用 bus 方面主动了，而该由每一个 driver 本身去 bus 上面登记，或者说挂牌。具体到 USB 子系统，每一个 USB 设备的驱动程序都会对应一个 struct usb_driver 结构，其中有一个 struct device_driver driver 成员，USB core 为每一个设备驱动准备了一个函数，让它把自己的这个 struct device_driver driver 插入到 usb_bus_type 中的 drivers 链表中去。而这个函数正是我们此前看到的 usb_register。而与之对应的 usb_deregister 所从事的正是与之相反的工作，把这个结构体从 drivers 链表中删除。

而 struct bus_type 结构的 match 函数在 USB 子系统里就是 usb_device_match 函数，它充当了一个红娘的角色，在 USB 总线的 USB 设备和 USB 驱动之间牵线搭桥，类似于交大 BBS 上的鹊桥版，虽然它们上面的条件都琳琅满目的，但明显这里 match 的条件不是那么的苛刻，要更为实际些。

可以说，USB core 的确是用心良苦，为每一个 USB 设备驱动做足了功课，正因为如此，作为一个实际的 USB 设备驱动，它在初始化阶段所要做的事情就很少，很简单了，直接调用 usb_register 即可。事实上，没有人是理所当然应该为你做什么的，但 USB core 这么做了。所以每一个写 USB 设备驱动的人应该铭记，USB 设备驱动绝不是一个人在工作，在他身后，是 USB core 所提供的默默无闻又不可或缺的支持。

设备模型之外，对于驱动程序的开发者来说，有三样东西是不可缺少的：第一是协议或标准的 spec，也就是规范，比如 usb 协议规范；第二是硬件的 datasheet，即你的驱动要支持的硬件的手册；第三就是内核里类似驱动的源代码，比如你要写触摸屏驱动的话，就可以参考内核里已经有的一些触摸屏驱动。

spec、datasheet、内核源代码这三样东西对于每个开发设备驱动的人来说都是再寻常不过了，但正是因为它们的普通，所以在很多人眼里都被归为被忽视的群体。于是大家开发驱动的过程中，遇到问题的时候首先想到的可能还是“问问牛人怎么解决吧”、“旁边要是有个牛人该多好”，因为牛人的稀有，所以知道牛人的价值，而又因为 spec、datasheet 和内核源代码的唾手可得，所以常

常体会不到它们在解决问题时的重要性。

当然我并不是贬低牛人的价值，宣扬依赖牛人不好，如果你很幸运身边真就有牛人这种稀缺资源的话，自然是要好好利用，也可以少走很多弯路，节省很多摸索的时间。只是人生不如意十之八九，多数人还是没有这份幸运的，所以与其遍寻牛人讨教，不如多依赖自己，多利用自己身边有的资源去寻找解决问题的途径。

对这三样看似普通的东西，关键在于很好的去利用，而不是拥有。就说 USB 吧，USB 驱动和 USB 设备如何进行交流，交流的方式，交流过程中出现了什么问题是什么引起的等等都在 USB spec 里有描述，而你的 USB 设备支持多少种配置包含多少端点只有设备的 datasheet 才知道。协议的 spec 和设备的 datasheet 是最好的参考资料，驱动开发调试中出现的问题绝大部分都能在它们的某个角落里找到答案。而内核中类似设备的驱动源代码是最好的模版，对很多硬件设备，你都可以内核找到同种设备的驱动代码进行参考实现，甚至于可以拷贝或共享大部分的代码，只进行局部的修改，比如说位于 drivers/input/touchscreen 目录下的各个触摸屏驱动，它们之间的代码很多都是类似的甚至是相同的。

如果你不仅仅只是打算写驱动，而是还想阅读内核中实现某种总线、设备的源代码，钻研它们的实现机制，那协议的 spec 就尤为重要，它们在代码里的体现无处不在，你需要在阅读代码前就对协议规范有个整体的理解。形象点说，spec 是理论基础，内核代码是具体实现，理论懂了，看代码就和看故事会差不多了。

Java 7 新 I/O 特性解析

Java 7 提供了一个新 API 访问文件系统，但除此之外，JSR 203 (NIO.2) 还包含其它很多新特性，这个新版本的确新增了很多改善 I/O 编程的类，本文将介绍下面的新特性：

1. SeekableByteChannel：随机访问通道；
2. MulticastChannel：允许 IP 多播的通道；
3. NetworkChannel：新的网络通道超级接口；
4. 异步 I/O API：新的 API 使 I/O 操作可以异步进行。

SeekableByteChannel

首先，Java 7 包括新的 ByteChannel – SeekableByteChannel，这个通道维护当前的位置，你可以读写该位置，并允许随机访问。使用这个类型的通道，你可以添加多个线程读/写在不同位置相同的线程。

```
SeekableByteChannel channel1 = Paths.get("Path to file").newByteChannel(); //Simply  
READ
```

```
SeekableByteChannel channel2 = Paths.get("Path to  
file").newByteChannel(StandardOpenOption.READ, StandardOpenOption.WRITE); //READ  
and WRITE
```

你可以使用下面这些方法操作位置和通道的大小。

1. long position()：返回目前的位置；
2. long size()：返回通道连接实体的当前大小，如通道连接的文件大小；

3. position(long newPosition): 移动当前位置到某个地方;
4. truncate(long size): 根据给定大小截断实体。

position()和truncate()方法简单地返回当前通道, 允许链式调用。

现在 FileChannel 实现了新的接口, 使用所有 FileChannel 你都可以实现随机访问, 当然你可以用它读取一个文件:

```
SeekableByteChannel channel = null;
try {
    channel = Paths.get("Path to file").newByteChannel(StandardOpenOption.READ);
    ByteBuffer buf = ByteBuffer.allocate(4096);

    System.out.println("File size: " + channel.size());

    String encoding = System.getProperty("file.encoding");

    while (channel.read(buf) > 0) {
        buf.rewind();

        byte[] bytearray = new byte[bytebuff.remaining()];
        buf.get(bytearray);
        System.out.print(new String(bytearray));

        buf.flip();

        System.out.println("Current position : " + channel.position());
    }
} catch (IOException e) {
    System.out.println("Expection when reading : " + e.getMessage());
    e.printStackTrace();
} finally {
    if (sbc != null){
        channel.close();
    }
}
```

```
}  
}
```

MulticastChannel

这个新的接口允许开启 IP 多播，因此你可以向一个完整的组发送和接收 IP 数据报。多播实现了直接绑定本地多播设备，这个接口是通过 `DatagramChannel` 和 `AsynchronousDatagramChannel` 实现的。

下面是从 Javadoc 中摘取的一个打开 `DatagramChannel` 的简单示例：

```
NetworkInterface networkInterface = NetworkInterface.getByName("hme0");  
  
DatagramChannel dc = DatagramChannel.open(StandardProtocolFamily.INET)  
    .setOption(StandardSocketOption.SO_REUSEADDR, true)  
    .bind(new InetSocketAddress(5000))  
    .setOption(StandardSocketOption.IP_MULTICAST_IF, networkInterface);
```

```
InetAddress group = InetAddress.getByName("225.4.5.6");
```

```
MembershipKey key = dc.join(group, networkInterface);
```

你可以使用以前经常使用的 `DatagramChannel`，但操作方式是多播了，因此你收到的是接口中所有的数据包，你发送的数据包会发到所有组。

NetworkChannel

现在所有网络通道都实现了新的 `NetworkChannel` 接口，你可以轻松绑定套接字管道，设置和查询套接字选项，此外，套接字选项也被扩展了，因此你可以使用操作系统特定的选项，对于高性能服务器这非常有用。

异步 I/O

现在我们介绍最重要的新特性：异步 I/O API，从它的名字我们就知道它有什么功能了，这个新的通道为套接字和文件提供了异步操作。

当然，所有操作都是非阻塞的，但对所有异步通道，你也可以执行阻塞操作，所有异步 I/O 操作都有下列两种形式中的一种：

- 1、第一个返回 `java.util.concurrent.Future`，代表等待结果，你可以使用 `Future` 特性等待 I/O 操作结束；
- 2、第二个是使用 `CompletionHandler` 创建的，当操作结束时，如回调系统，调用这个处理程序。

下面是它们的一些例子，首先来看看使用 `Future` 的例子：

```
AsynchronousFileChannel channel = AsynchronousFileChannel.open(Paths.get("Path to file"));
```

```
ByteBuffer buffer = ByteBuffer.allocate(capacity);
```

```
Future result = channel.read(buffer, 100); //Read capacity bytes from the file starting at  
position 100
```

```
boolean done = result.isDone(); //Indicate if the result is already terminated
```

你也可以等待结束：

```
int bytesRead = result.get();
```

或等待超时：

```
int bytesRead = result.get(10, TimeUnit.SECONDS); //Wait at most 10 seconds on the result
```

再来看看使用 CompletionHandler 的例子：

```
Future result = channel.read(buffer, 100, null, new CompletionHandler(){
```

```
    public void completed(Integer result, Object attachment){
```

```
        //Compute the result
```

```
    }
```

```
    public void failed(Throwable exception, Object attachment){
```

```
        //Answer to the fail
```

```
    }
```

```
});
```

正如你所看到的，你可以给操作一个附件，在操作末尾给 CompletionHandler，当然，你可以把 null 当作一个附件提供，你可以传递任何你想传递的，如用于 AsynchronousSocketChannel 的 Connection，或用于读操作的 ByteBuffer。

```
Future result = channel.read(buffer, 100, buffer, new CompletionHandler(){
```

```
    public void completed(Integer result, ByteBuffer buffer){
```

```
        //Compute the result
```

```
    }
```

```
    public void failed(Throwable exception, ByteBuffer buffer){
```

```
        //Answer to the fail
```

```
    }
```

```
});
```


正如你所看到的，CompletionHandle 也提供 Future 元素表示等待结果，因此你可以合并这两个格式。

下面是 NIO.2 中的所有异步通道：

1. AsynchronousFileChannel: 读写文件的异步通道，这个通道没有全局位置，因此每个读写操作都需要一个位置，你可以使用不同的线程同时访问文件的不同部分，当你打开这个通道时，必须指定 READ 或 WRITE 选项；
2. AsynchronousSocketChannel: 用于套接字的一个简单异步通道，连接、读/写、分散/聚集方法全都是异步的，读/写方法支持超时；
3. AsynchronousServerSocketChannel: 用于 ServerSocket 的异步通道，accept()方法是异步的，当连接被接受时，调用 CompletionHandler，这种连接的结果是一个 AsynchronousSocketChannel；
4. AsynchronousDatagramChannel: 用于数据报套接字的通道，读/写（连接）和接收/发送（无连接）方法是异步的。

分组

当你使用 AsynchronousChannels 时，有线程调用完整的处理程序，这些线程被绑定到一个 AsynchronousChannelGroup 组，这个组包含一个线程池，它封装了所有线程共享的资源，你可以使用线程池来调用这些组，AsynchronousFileChannel 可以使用它自己的组创建，将 ExecutorService 作为一个参数传递给 open()方法，在 open 方法中，通道是使用 AsynchronousChannelGroup 创建的，如果你不给它一个组，或传递一个 NULL，它就会使用默认组。通道被认为是属于组的，因此，如果组关闭了，通道也就关闭了。

你可以使用 ThreadFactory 创建一个组：

```
ThreadFactory myThreadFactory = Executors.defaultThreadFactory();
```

```
AsynchronousChannelGroup channelGroup =  
AsynchronousChannelGroup.withFixedThreadPool(25, threadFactory);
```

或使用一个 ExecutorService：

```
ExecutorService service = Executors.newFixedThreadPool(25);
```

```
AsynchronousChannelGroup channelGroup =  
AsynchronousChannelGroup.withThreadPool(service);
```

而且你可以很容易地使用它：

```
AsynchronousSocketChannel socketChannel =  
AsynchronousSocketChannel.open(channelGroup);
```

你可以使用在组上使用 shutdown()方法关闭组，关闭之后，你就不能使用这个组创建更多的通道，当所有通道关闭后，组也终止了，处理程序结束，资源也释放了。

当你使用任何类型的池和 CompletionHandler 时，你必须要注意一点，不要在 CompletionHandler 内使用阻塞或长时间操作，如果所有线程都被阻塞，整个应用程序都会被阻塞

掉。如果你有自定义或缓存的线程池, 它会使队列无限制地增长, 最终导致 OutOfMemoryError。

我想我把新的异步 I/O API 涵盖的内容讲得差不多了, 当然这不是一两句话可以说清楚的, 也不是每一个人都会使用到它们, 但在某些时候它确实很有用, Java 支持这种 I/O 操作终归是一件好事。如果我的代码中有什么错误我表示道歉, 因为我也刚刚才接触。

原文出处: www.baptiste-wicht.com/2010/04/java-7-new-io-features-asynchronous-operations-multicasting-random-access-with-jsr-203-nio-2/

原文名: Java 7 : New I/O features (Asynchronous operations, multicasting, random access) with JSR 203 (NIO.2)

作者: Baptiste Wicht

翻译: 黄永兵 [Email: swnuhyb001@163.com](mailto:swnuhyb001@163.com)

邮件服务 postfix 详解

ChinaUnix 网友: Sandy

电子邮件系统与其他 Internet 服务相同, 电子邮件是基于客户/服务器模式的. 对于一个完整的电子邮件系统而言, 它主要由一下三部分构件组成...

1. 用户代理: 用户代理 (User Agent, 缩写 UA) 就是用户与电子邮件系统的接口, 在大多数情况下它就是在邮件客户端上运行的程序, 主要负责将邮件发送到邮件服务器和从邮件服务器上接收邮件. 目前主流的用户代理主要由 Microsoft 公司的 Outlook 和国产的 Foxmail 等...

2. 邮件服务器: 主要就是发送和接收邮件.. 根据用途不同, 可将邮件服务器分为发送邮件服务器 (SMTP 服务器) 和接收邮件服务器 (POP3 服务器或 IMAP4 服务器).

3. 电子邮件使用的协议:

SMTP 协议: SMTP 即简单邮件传输协议, 它是一组用于源地址到目的地址传送邮件的规则, 由它来控制信件的中转方式.

POP3 协议: POP3 即邮局协议的第三个版本, 它规定怎样将个人计算机连接到 Internet 的邮件服务器和下载电子邮件的协议. 它是 Internet 电子邮件的第一个离线协议标准. POP3 允许从服务器上把邮件存储到本地主机即自己的计算机上, 同时删除保存在邮件服务器上的邮件.

IMAP4 协议: IMAP4 即 Internet 信息访问协议的第四个版本, 是用于从本地服务器上访问电子邮件的协议, 它是一个客户/服务器模型协议, 用户的电子邮件由服务器负责接收保存, 用户可以通过浏览器信件头来决定是否要下载此信. 用户也可以在服务器上创建或更改文件夹或邮箱, 删除信件或检索信件特定部分.

虽然 POP 和 IMAP 都是处理接收邮件的, 但两者在机制上却有所不同. 在用户访问电子邮件时, IMAP4 需要持续访问服务器, POP3 则是将信件保存在服务器上, 当用户阅读信件时, 所有内容都会被立即下载到用户的机器上. 因此可以把 IMAP4 看成是一个远程文件服务器, 而把 POP 看成是一个存储转发服务器. 就目前情况下看, POP3 的应用远比 IMAP4 广泛得多.

在 linux 平台中, 有许多邮件服务器可供选择, 但目前使用较多的是 sendmail 服务器, postfix 服务器, Qmail 服务器.

与 sendmail 相比,postfix 最被人称道的地方就在于其配置文件可读性很高.postfix 的主配置文件是/etc/postfix/main.cf,先去理解几个代名词...

MUA --> 邮件用户代理. Outlook. Foxmail. thunderbird. Evolution. Mutt

MTA --> 邮件传递代理.

MDA --> 邮件投递代理. Procmail. maildrop

MAA --> 邮件访问代理. imap. cryas-imap. dovecot

SASL --> 简单认证安全层.(只是给 SMTP 提供了一种认证功能)

MIME --> 多用途邮件 Internet 邮件扩展.

MRA --> 邮件检索代理和 MAA 性质一样.

首先我们去看看配置文件/etc/postfix/main.cf 中的选项的意义...

myhostname = mail.zzu.com --> 设置 postfix 的主机名称.

myorigin = \$myhostname --> 设置由本台邮件主机寄出的每封邮件的邮件头中 mail

from 的地址,postfix 默认使用本地主机名作为 myorigin 参数值,建议用户将 myorigin 参数设置为本地邮件主机的域名.

mydomain = zzu.com --> 指定该主机的域名.

inet_interfaces --> 默认情况下此值被设置为 localhost.这表明只能在本地邮件主机上寄信.如果邮件主机上有多个网络接口.不过,通常是将所有的网络接口 开放,以便接受从任何接口来的邮件..设置为 all mydestination --> 只有当发来的邮件的收件人地址与该参数值相匹配时,postfix 才会

将邮件接收下来.

mynetworks --> 可将该参数值设置为所信任的某台主机的 IP 地址,也可设置为所信任的某个 IP 子网或多个 IP 子网(用","分隔),这里将值设置为 192.168.1.0/24,表示这台机器只转发子网 192.168.1.0/24 中客户机所发来的邮件,拒绝为其它子网转发邮件.

mynetworks-style --> 主要设置可转发邮件的网络的方式.主要由三种:class. subnet.

host

relay_domains --> 参数则是针对邮件来源的域名或主机名来设置的.列如:将参数值设置为 zzu.com,则表示任何由域 zzu.com 发来的邮件都会被认为是信任的,postfix 会将这些邮件进行转发..

虚拟别名域的配置

使用虚拟别名域,可以将发给虚拟域的邮件实际投递到真实 域的用户邮箱中;可以实现群组邮递的功能,即指定一个虚拟邮件地址,任何人发给这个邮件地址都将由邮件服务器自动转发到真实域中的一组用户邮箱中.这里的 虚拟域可以是实际并不存在的域,而真实域即可以是本地域(即 main.cf 文件中的 mydestination 参数值中列出的域),也可以是远程域或 Internet 中的域.虚拟域是真实域的一个别名,实际上是通过一个虚拟别名表(virtual),实现了虚拟域的邮件地址到真实域的邮件地址的重定向..

配置 postfix 必须关闭 sendmail 服务....

```
#service sendmail stop
```

```
#chkconfig sendmail off
```

```
#yum -y install postfix
```

在实际应用中要实现虚拟别名域,必须按以下步骤进行.编辑/etc/postfix/main.cf 进行如下定义...

virtual_alias_domains = zzu.com

virtual_alias_maps = hash:/etc/postfix/virtual

这里参数 virtual_alias_domain 定义虚拟别名域的名称,virtual_alias_maps 定义虚拟别名域定义的文件路径...

编辑/etc/postfix/virtual

@example.com @zzu.com -->发给虚拟域 example.com 的邮件实际投到真实的本地域 @zzu.com

admin@example.com sandy,mary,natasha@sina.com -->发给虚拟用户 admin 的邮件会转发给本地用户 sandy 和 mary 或者 internet 的用户 natasha.

修改完后执行下面命令生效...

#postmap /etc/postfix/virtual -->用来将文件/etc/postfix/virtual 生产 postfix 可以读取的数据库文件/etc/postfix/virtual.db

#postfix reload -->重新加载 postfix 的主配置文件..

用户的别名的配置...

本地 linux 系统中有个用户 sandy,并且在 Internet 中有个 Email:oranix@sina.com,为他设置两个别名,分别是 haha,hehe.编辑/etc/aliases

haha: sandy

hehe: sandy,oranix@sina.com

还必须编辑/etc/postfix/main.cf 文件...

alias_maps = hash:/etc/aliases -->用来指定含有用户别名定义的文件路径,alias_database 用来指定别名表数据库文件路径.

alias_database = hash:/etc/aliases -->用来指定别名表数据库文件路径.

注意配置文件中等号前后一定要有空格.修改完配置文件后要使其生效,使用下面的命令...

#postalias /etc/aliases -->用来将文件/etc/aliases 生成 postfix 可以读取的数据库文件 /etc/aliases.db

#postfix reload

用户别名可以实现邮件列表的功能,但是只有 root 用户才能修改 aliases 文件...

ARM-Linux 使用 SD 卡根文件系统

ChinaUnix 网友: bluedrum

因此指导学员在 arm-Linux 使用 JPT-7 模块来跑 GPS 应用.Nand Flash 空间太小,而且所用的板经常烧不了根文件系统.所以决定用 SD 卡来跑根文件系统。

1.PC 机上格式化 SD 卡

在桌面的 PC 机上用 SD 读卡器操作 SD 卡:

一般情况下 PC 机上第一个 U 盘整体设备结点是/dev/sda,第二个是 /dev/sdb,在 RHEL5 下它会被自动 mount 到/media/disk 和 /media/disk_1 目录。

U 盘/dev/sdb 上第一个分区是 /dev/sdb1,第二是/dev/sdb2 依此类推。

根文件系统采用符号链接等特性,用 FAT32 是不行的,这里直接采用标准的 ext3 的文件系统.在实测时,把 U 盘整个做一个分区做 ext3 根文件系统.总是出不少问题(可能步骤也不对),因此按网上推荐的,做二个分区,第一个分区采用 vfat 格式,第二个分区才采用 ext3 的格式。

1.1 用 fdisk 分区

执行 **fdisk /dev/sdb**

fdisk 有如下常用选项

1. 输入 m 显示所有命令列示。
2. 输入 p 显示硬盘分割情形。
3. 输入 a 设定硬盘启动区。
4. 输入 n 设定新的硬盘分割区。
 - 4.1. 输入 e 硬盘为[扩展]分割区(extend)。
 - 4.2. 输入 p 硬盘为[首要]分割区(primary)。
5. 输入 t 改变硬盘分割区属性。
6. 输入 d 删除硬盘分割区属性。
7. 输入 q 结束不存入硬盘分割区属性。
8. 输入 w 结束并写入硬盘分割区属性

如果以前 U 盘有分区,需要输入 d 命令来依次删除分区,以下执行两次 n 命令创建一个 400M 的 FAT 分区,以及把剩下的分区设为 ext3,最后用 w 命令把结果保存下来。

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1020, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1020, default 1020): +400M

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (202-1020, default 202):
Using default value 202
Last cylinder or +size or +sizeM or +sizeK (202-1020, default 1020):
Using default value 1020

Command (m for help): p
Disk /dev/sdb: 2041 MB, 2041577472 bytes
```


63 heads, 62 sectors/track, 1020 cylinders
Units = cylinders of 3906 * 512 = 1999872 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		1	201	392522	b	W95 FAT32
/dev/sdb2		202	1020	1599507	83	Linux

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: 设备或资源忙.

The kernel still uses the old table.

The new table will be used at the next reboot.

WARNING: If you have created or modified any DOS 6.x partitions, please see the fdisk manual page for additional information.

Syncing disks.

1.2 格式化分区

`mkfs.vfat /dev/sda1` #第一个分区格式化成 fat32

`mkfs.ext3 /dev/sda2` #第二个分区格式化成 fat32

可以用

`tune2fs -l /dev/sda2` 来检验分区类型

1.3 拷贝根文件系统

`mount /dev/sda2 /mnt/sdcard` #挂载 SD 卡

`cp -ra /home/hxy/rootfs/* /mnt/sdcard` #把根文件系统拷到 SD 卡

2.编译 ARM Linux 内核

2.1 修改内核配置

确保 ext3 的文件系统支持,和 SD 卡驱动都被静态编译到内核,我使用的是 Linux 2.6.29.

在内核源码目录执行 `make menuconfig`

其中 ext3 在 File system 下.成功的配置如下。

<*> Ext3 journalling file system support

[*] Ext3 extended attributes

[*] Ext3 POSIX Access Control Lists

[*] Ext3 Security Labels

SD 卡驱动支持在 Device Drivers ->MMC/SD/SDIO card support 下,成功的配置如下

```
--- MMC/SD/SDIO card support
[*] MMC debugging
[] Allow unsafe resume (DANGEROUS)
*** MMC/SD/SDIO Card Drivers ***
<*> MMC block device driver
[*] Use bounce buffer for simple hosts
<*> SDIO UART/GPS class support
<*> MMC host test driver
*** MMC/SD/SDIO Host Controller Drivers ***
<*> Secure Digital Host Controller Interface support
<*> Samsung S3C SD/MMC Card Interface
support
```

编译内核 make zImage

2.2 在 ARM-linux 下的测试.

在一个已经正常启动的 ARM-LINUX 下,插入 SD 卡.出现如下提示,表示 SD 卡已经安装上

```
[root: /]# s3c2440-sdi s3c2440-sdi: running at 0kHz (requested: 0kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
mmc0: new SD card at address 0002
mmcblk0: mmc0:0002 00000 1.90 GiB
mmcblk0: p1 p2
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will be case
sensitive!
```

但是最后一句在提示 FAT 有不识别的 IO 字符集,导致第一个 VFAT 分区没有自动创建设备结点.用如下命令只看到 mmcblk0p2 的结点.(以下操作均是在开发板上进行)

```
[root: /]# ls -l /dev/mmc*
brw-rw---- 1 root root 179, 0 Sep 22 10:48 /dev/mmcblk0
brw-rw---- 1 root root 179, 2 Sep 22 10:48 /dev/mmcblk0p2
```

测试 ext3 分区

```
mount -t ext3 /dev/mmcbk0p2 /mnt
```

在/mnt 上可以读写文件.表示 ext3 分区正常.

手动建立第一个分区的结点.

```
mknod /dev/mmcbk0p1 b 179 1
```

测试 vfat 分区

```
mount -t vfat /dev/mmcbk0p1 /mnt
```

在/mnt 上可以读写文件.表示 vfat 分区正常.如果想自动创建个设备结点,可以修改/etc/fstab 来使用这个分区

3.使用 SD 卡分区作为根文件

修改 LINUX 启动参数.我用的是 u-boot .因此在 u-boot shell 用 set bootargs 命令来设置 Linux 启动参数。

成功的启动参数是 `noinitrd root=179:2 rw console=ttySAC0`

其中 179 和 2 是 sd 卡上 ext3 分区的主设备和从设备号。

用 set bootargs "noinitrd root=179:2 rw console=ttySAC0 "设置后.重启内核.即可用 ext3 分区作为根文件系统。

OpenSSL 库中 des 加密函数库的使用

ChinaUnix 网友: harrypolt2008

DES 是一种可以实现数据加密和解密的对称算法, 在 OpenSSL 中有 DES 算法的具体实现, 由于是开源代码, 你不需要任何借口就可以使用它们, 当然最好你也遵守和它一样的开源协议。在很多的 Linux 系统中已经安装了 openssl 函数库, 如果你的 Linux 系统没有安装 OpenSSL 函数库的话, 你需要安装它们, 这样你才可以使用此函数库。当然, 如果你需要查看它们的使用方法的话, 你还需要安装此函数的开发文档, 这样你就可以利用 man 手册来方便的了解此函数库了。下面是本人利用此 DES 算法 API 实现的一个简单的数据加密和解密测试程序, 希望能对学习如何使用此 DES 函数库的朋友们有所帮助, 错误之处, 请不吝赐教。后面附加了一个简单的 makefile 文件。

```
/*main 文件*/
#include <stdio.h>
#include <openssl/des.h> /*DES 算法头文件*/

#define DEBUG_PRINT

/*初始化需要加密的数据*/
void init_input(const_DES_cblock input)
{
```

```
input[0] = 0;
input[1] = 1;
input[2] = 2;
input[3] = 3;
input[4] = 4;
input[5] = 5;
input[6] = 6;
input[7] = 7;
}

int main(void)
{
    int ret = 0;
    int i = 0;
    DES_cblock key; /*密钥*/
    DES_key_schedule key_schedule; /*将密钥转化保存的数据结构，用于加密*/
    DES_key_schedule key_schedule1; /*解密时使用*/
    const_DES_cblock input; /*需要加密的数据*/
    DES_cblock output; /*加密后的数据*/

    init_input(input); /*初始化数据*/
    memset(output, 0, 8); /*清零*/

    ret = DES_random_key(&key); /*获得随机密钥*/
    ret = DES_set_key_checked(&key, &key_schedule); /*转化密钥*/

    DES_ecb_encrypt(&input, &output, &key_schedule, 1); /*加密数据*/

    memset(input, 0, 8); /*清零*/
    ret = DES_set_key_checked(&key, &key_schedule1); /*转换密钥*/
    DES_ecb_encrypt(&output, &input, &key_schedule1, 0); /*解密*/

    return 0;
}

/*makefile*/
CC=gcc
CFLAGS=-lssl
```

all: main

main: main.c

\$(CC) -o \$@ \$^ \$(CFLAGS)

clean:

rm -rf main

Linux 系统管理员应该知道的 20 个系统监控工具

ChinaUnix 网友: imlidapeng

需要监控 Linux 服务器系统性能吗? 尝试下面这些系统内置或附件的工具吧。大多数 Linux 发行版本都装备了大量 的监控工具。这些工具提供了能用作取得相关信息和系统活动的量度指标。你能使用这些工具发现造成性能问题可能原因。此次讨论到的工具只是分析和调试服务器 下面问题时最基本工具中的一部分。

1.找出瓶颈

2.硬盘（存储）瓶颈

3.CPU 及内存瓶颈

4.网络瓶颈

#1: top - 进程活动

top 提供一个当前运行系统实时动态的视图，也就是正在运行进程。在默认情况下，显示系统中 CPU 使用率最高的任务，并每 5 秒钟刷新一次。

```
top - 04:14:53 up 1 day, 20:07, 5 users, load average: 0.53, 0.69, 0.55
tasks: 187 total, 2 running, 184 sleeping, 0 stopped, 1 zombie
%cpu: 4.6%us, 0.5%sy, 0.0%ni, 94.8%id, 0.1%wa, 0.0%hi, 0.0%st
Mem: 8299944k total, 8820764k used, 279160k free, 221276k buffers
Swap: 195180k total, 2976k used, 194012k free, 6454792k cached
```

PID	USER	PR	NI	VT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8352	vivek	20	0	240m	73m	29m	S	10	0.9	6:21.26	liferay-bin
4575	vivek	20	0	227m	13m	10m	S	5	1.5	86:26.28	deluge
29831	vivek	20	0	766m	47m	34m	S	3	5.8	43:02.18	firefox-bin
6873	root	20	0	366m	95m	17m	S	2	1.2	88:44.31	Xorg
7268	vivek	20	0	31412	5240	3820	S	1	0.1	10:26.03	pulseaudio
12592	root	15	-5	0	0	0	S	1	0.0	10:54.52	ntos_wq
32484	vivek	20	0	84512	26m	11m	S	0	0.3	0:07.67	gnome-terminal
1	root	20	0	1980	884	452	S	0	0.0	0:01.69	init
2	root	15	-5	0	0	0	S	0	0.0	0:00.00	kthreadd
3	root	RT	-5	0	0	0	S	0	0.0	0:00.02	migration/0
4	root	15	-5	0	0	0	R	0	0.0	0:27.71	ksoftirqd/0
5	root	RT	-5	0	0	0	S	0	0.0	0:00.00	watchdog/0
6	root	RT	-5	0	0	0	S	0	0.0	0:00.01	migration/1
7	root	15	-5	0	0	0	S	0	0.0	0:06.38	ksoftirqd/1
8	root	RT	-5	0	0	0	S	0	0.0	0:00.00	watchdog/1
9	root	RT	-5	0	0	0	S	0	0.0	0:00.02	migration/2
10	root	15	-5	0	0	0	S	0	0.0	0:05.59	ksoftirqd/2
11	root	RT	-5	0	0	0	S	0	0.0	0:00.00	watchdog/2
12	root	RT	-5	0	0	0	S	0	0.0	0:00.03	migration/3
13	root	15	-5	0	0	0	S	0	0.0	0:05.83	ksoftirqd/3
14	root	RT	-5	0	0	0	S	0	0.0	0:00.00	watchdog/3
15	root	15	-5	0	0	0	S	0	0.0	0:00.27	events/0
16	root	15	-5	0	0	0	S	0	0.0	0:00.52	events/1
17	root	15	-5	0	0	0	S	0	0.0	0:00.44	events/2
18	root	15	-5	0	0	0	S	0	0.0	0:00.50	events/3
19	root	15	-5	0	0	0	S	0	0.0	0:00.01	khelper
61	root	15	-5	0	0	0	S	0	0.0	0:00.00	kintegrityd/0
62	root	15	-5	0	0	0	S	0	0.0	0:00.00	kintegrityd/1
63	root	15	-5	0	0	0	S	0	0.0	0:00.00	kintegrityd/2
64	root	15	-5	0	0	0	S	0	0.0	0:00.00	kintegrityd/3
66	root	15	-5	0	0	0	S	0	0.0	0:00.65	kblocks/0
67	root	15	-5	0	0	0	S	0	0.0	0:00.16	kblocks/1
68	root	15	-5	0	0	0	S	0	0.0	0:00.23	kblocks/2
69	root	15	-5	0	0	0	S	0	0.0	0:00.11	kblocks/3
71	root	15	-5	0	0	0	S	0	0.0	0:00.00	kacpid
72	root	15	-5	0	0	0	S	0	0.0	0:00.00	kacpi_notify
153	root	15	-5	0	0	0	S	0	0.0	0:00.00	cqueue
157	root	15	-5	0	0	0	S	0	0.0	0:00.01	kserial
231	root	15	-5	0	0	0	S	0	0.0	0:02.23	kswapd0

图 01.Linux top 命令

常用热键

热键 用途

- t 显示摘要信息开关.
- m 显示内存信息开关.
- A 分类显示系统不同资源的使用大户。有助于快速识别系统中资源消耗多的任务。
- f 添加删除所要显示栏位.
- o 调整所要显示栏位的顺序.
- r 调整一个正在运行的进程 Nice 值.
- k 结束一个正在运行的进程.
- z 彩色/黑白显示开关

相关链接: [How do I Find Out Linux CPU Utilization?](#)

译者推荐链接: [Linux 系统管理员必备工具系列之 top \(原创\)](#)

#2:vmstat -系统活动、硬件及系统信息

使用 vmstat 命令可以得到关于进程、内存、内存分页、堵塞 IO、traps 及 CPU 活动的信息。

vmstat 3

输出样例:

```
procs -----memory----- --swap-- ----io---- --system-- ----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0  0 2540988 522188 5130400 0 0  2 32  4  2 4 1 96 0 0
1 0  0 2540988 522188 5130400 0 0  0 720 1199 665 1 0 99 0 0
0 0  0 2540956 522188 5130400 0 0  0  0 1151 1569 4 1 95 0 0
0 0  0 2540956 522188 5130500 0 0  0  6 1117 439 1 0 99 0 0
0 0  0 2540940 522188 5130512 0 0  0 536 1189 932 1 0 98 0 0
0 0  0 2538444 522188 5130588 0 0  0  0 1187 1417 4 1 96 0 0
0 0  0 2490060 522188 5130640 0 0  0 18 1253 1123 5 1 94 0 0
```

显示内存使用详细信息

vmstat -m

显示内存活动/不活动的信息

vmstat -a

相关链接: [How do I find out Linux Resource utilization to detect system bottlenecks?](#)

译者推荐链接: [Linux 系统管理员必备工具系列之 vmstat \(原创\)](#)

#3:w - 显示谁已登录，他们正在做什么？

w 命令显示系统当前用户及其运行进程的信息。

```
# w username
```

```
# w vivek
```

输出样例:

```
17:58:47 up 5 days, 20:28, 2 users, load average: 0.36, 0.26, 0.24
```

```
USER  TTY  FROM      LOGIN@  IDLE  JCPU  PCPU  WHAT
```

```
root  pts/0  10.1.3.145  14:55  5.00s  0.04s  0.02s  vim /etc/resolv.conf
```

```
root  pts/1  10.1.3.145  17:43  0.00s  0.03s  0.00s  w
```

#4: uptime - 告诉系统已经运行了多久?

uptime 命令过去只显示系统运行多久。现在，可以显示系统运行多久、当前有多少的用户登录、在过去的 1，5，15 分钟里平均负载时多少。

```
# uptime
```

输入样例:

```
18:02:41 up 41 days, 23:42, 1 user, load average: 0.00, 0.00, 0.00
```

1 可以被认为是最优的负载值。负载是会随着系统不同改变得。单 CPU 系统 1-3 和 SMP 系统 6-10 都是可能接受的。

#5: ps - 显示进程

ps 命令显示当前运行进程的快照。使用-A 或-e 显示所有进程。

```
# ps -A
```

输出样例:

```
PID TTY      TIME CMD
```

```
1 ?      00:00:02 init
```

```
2 ?      00:00:02 migration/0
```

```
3 ?      00:00:01 ksoftirqd/0
```

```
4 ?      00:00:00 watchdog/0
```

```
5 ?      00:00:00 migration/1
```

```
6 ?      00:00:15 ksoftirqd/1
```

```
....
```

```
.....
```

```
4881 ?    00:53:28 java
```

```
4885 tty1   00:00:00 mingetty
```

```
4886 tty2   00:00:00 mingetty
```

```
4887 tty3 00:00:00 mingetty
4888 tty4 00:00:00 mingetty
4891 tty5 00:00:00 mingetty
4892 tty6 00:00:00 mingetty
4893 ttyS1 00:00:00 agetty
12853 ? 00:00:00 cifsoplockd
12854 ? 00:00:00 cifsnotifyd
14231 ? 00:10:34 lighttpd
14232 ? 00:00:00 php-cgi
54981 pts/0 00:00:00 vim
55465 ? 00:00:00 php-cgi
55546 ? 00:00:00 bind9-snmp-stat
55704 pts/1 00:00:00 ps
```

ps 与 top 非常相似，但 ps 提供更多的信息。

输出长格式

```
# ps -Al
```

输出附加全格式（显示进程在执行时传入的参数）

```
# ps -AlF
```

显示进程结构

```
# ps -AlFH
```

在进程后显示线程

```
# ps -AlLm
```

打印服务器上所有进程

```
# ps ax
```

```
# ps axu
```

打印进程树

```
# ps -ejH
```

```
# ps axjf
```

```
# pstree
```

打印安全信息

```
# ps -eo euser,ruser,suser,fuser,f,comm,label
```

```
# ps axZ
```

ps -eM

查看使用 Vivek 用户名运行的进程

ps -U vivek -u vivek u

设置自定义输出格式

ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm

ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm

ps -eopid,tt,user,fname,tmout,f,wchan

只显示 Lighttpd 的进程 ID

ps -C lighttpd -o pid=

或者

pgrep lighttpd

或者

pgrep -u vivek php-cgi

显示 PID 为 55977 的进程名称

ps -p 55977 -o comm=

找出消耗内存最多的前 10 名进程

ps -auxf | sort -nr -k 4 | head -10

找出使用 CPU 最多的前 10 名进程

ps -auxf | sort -nr -k 3 | head -10

#6:free - 内存使用情况

free 命令显示系统中空闲的、已用的物理内存及 swap 内存,及被内核使用的 buffer。

free

输出样例:

	total	used	free	shared	buffers	cached
--	-------	------	------	--------	---------	--------

Mem:	12302896	9739664	2563232	0	523124	5154740
------	----------	---------	---------	---	--------	---------

-/+ buffers/cache:	4061800	8241096				
--------------------	---------	---------	--	--	--	--

Swap:	1052248	0	1052248			
-------	---------	---	---------	--	--	--

相关链接:

1. [Linux Find Out Virtual Memory PAGESIZE](#)
2. [Linux Limit CPU Usage Per Process](#)
3. [How much RAM does my Ubuntu / Fedora Linux desktop PC have?](#)

#7:iostat - CPU 平均负载, 硬盘活动

iostat 命令可报告中央处理器（CPU）的统计信息，各种设备、分区及网络文件系统输入/输出的统计信息。

```
# iostat
```

输出样例：

Linux 2.6.18-128.1.14.el5 (www03.nixcraft.in) 06/26/2009

avg-cpu: %user %nice %system %iowait %steal %idle

3.50 0.09 0.51 0.03 0.00 95.86

Device: tps Blk_read/s Blk_wrtn/s Blk_read Blk_wrtn

sda 22.04 31.88 512.03 16193351 260102868

sda1 0.00 0.00 0.00 2166 180

sda2 22.04 31.87 512.03 16189010 260102688

sda3 0.00 0.00 0.00 1615 0

相关链接: [Linux Track NFS Directory / Disk I/O Stats](#)

#8:sar - 搜集和报告系统活动

sar 命令用来搜集、报告和储存系统活动信息。查看网路计数器，输入：

```
# sar -n DEV | more
```

显示最近 24 小时网络计数器

```
# sar -n DEV -f /var/log/sa/sa24 | more
```

你亦可以用 sar 显示实时情况

```
# sar 4 5
```

输出样例：

Linux 2.6.18-128.1.14.el5 (www03.nixcraft.in) 06/26/2009

	CPU	%user	%nice	%system	%iowait	%steal	%idle
--	-----	-------	-------	---------	---------	--------	-------

06:45:12 PM	all	2.00	0.00	0.22	0.00	0.00	97.78
-------------	-----	------	------	------	------	------	-------

06:45:20 PM	all	2.07	0.00	0.38	0.03	0.00	97.52
-------------	-----	------	------	------	------	------	-------

06:45:24 PM	all	0.94	0.00	0.28	0.00	0.00	98.78
-------------	-----	------	------	------	------	------	-------

06:45:28 PM	all	1.56	0.00	0.22	0.00	0.00	98.22
-------------	-----	------	------	------	------	------	-------

06:45:32 PM	all	3.53	0.00	0.25	0.03	0.00	96.19
-------------	-----	------	------	------	------	------	-------

Average:	all	2.02	0.00	0.27	0.01	0.00	97.70
----------	-----	------	------	------	------	------	-------

相关链接: [How to collect Linux system utilization data into a file](#)

#9:mpstat - 多处理器使用率

mpstat 命令可以显示所有可用处理器的使用情况, 处理器编号从 0 开始。mpstat -P ALL 显示每个处理器的平均使用率。

```
# mpstat -P ALL
```

输出样例:

Linux 2.6.18-128.1.14.el5 (www03.nixcraft.in) 06/26/2009

06:48:11 PM	CPU	%user	%nice	%sys	%iowait	%irq	%soft	%steal	%idle	intr/s
06:48:11 PM	all	3.50	0.09	0.34	0.03	0.01	0.17	0.00	95.86	1218.04
06:48:11 PM	0	3.44	0.08	0.31	0.02	0.00	0.12	0.00	96.04	1000.31
06:48:11 PM	1	3.10	0.08	0.32	0.09	0.02	0.11	0.00	96.28	34.93
06:48:11 PM	2	4.16	0.11	0.36	0.02	0.00	0.11	0.00	95.25	0.00
06:48:11 PM	3	3.77	0.11	0.38	0.03	0.01	0.24	0.00	95.46	44.80
06:48:11 PM	4	2.96	0.07	0.29	0.04	0.02	0.10	0.00	96.52	25.91
06:48:11 PM	5	3.26	0.08	0.28	0.03	0.01	0.10	0.00	96.23	14.98
06:48:11 PM	6	4.00	0.10	0.34	0.01	0.00	0.13	0.00	95.42	3.75
06:48:11 PM	7	3.30	0.11	0.39	0.03	0.01	0.46	0.00	95.69	76.89

相关链接: [Linux display each multiple SMP CPU processors utilization individually.](#)

#10: pmap - 进程的内存使用

pmap 命令可以显示进程的内存映射, 使用这个命令可以找出造成内存瓶颈的原因。

```
# pmap -d PID
```

显示 PID 为 47394 进程的内存信息。

```
# pmap -d 47394
```

输出样例:

```
47394: /usr/bin/php-cgi
Address      Kbytes Mode Offset      Device  Mapping
0000000000400000 2584 r-x-- 0000000000000000 008:00002 php-cgi
0000000000886000 140 rw--- 00000000000286000 008:00002 php-cgi
```

```
00000000008a9000    52 rw--- 00000000008a9000 000:00000 [ anon ]
0000000000aa8000    76 rw--- 00000000002a8000 008:00002 php-cgi
0000000000f678000  1980 rw--- 0000000000f678000 000:00000 [ anon ]
000000314a60000    112 r-x-- 0000000000000000 008:00002 ld-2.5.so
000000314a81b000     4 r---- 000000000001b000 008:00002 ld-2.5.so
000000314a81c000     4 rw--- 000000000001c000 008:00002 ld-2.5.so
000000314aa0000    1328 r-x-- 0000000000000000 008:00002 libc-2.5.so
000000314ab4c000   2048 ----- 000000000014c000 008:00002 libc-2.5.so
.....
.....
..
00002af8d48fd000     4 rw--- 0000000000006000 008:00002 xsl.so
00002af8d490c000    40 r-x-- 0000000000000000 008:00002 libnss_files-2.5.so
00002af8d4916000   2044 ----- 000000000000a000 008:00002 libnss_files-2.5.so
00002af8d4b15000     4 r---- 0000000000009000 008:00002 libnss_files-2.5.so
00002af8d4b16000     4 rw--- 000000000000a000 008:00002 libnss_files-2.5.so
00002af8d4b17000 768000 rw-s- 0000000000000000 000:00009 zero (deleted)
00007fffc95fe000    84 rw--- 00007fffffea000 000:00000 [ stack ]
fffffffff600000   8192 ----- 0000000000000000 000:00000 [ anon ]
mapped: 933712K  writeable/private: 4304K  shared: 768000K
```

最后一行非常重要:

* mapped: 933712K 内存映射所占空间大小

* writeable/private: 4304K 私有地址空间大小

* shared: 768000K 共享地址空间大小

相关链接: [Linux find the memory used by a program / process using pmap command](#)

#11 和#12: netstat 和 ss - 网络相关信息

netstat 可以显示网络链接、路由表信息、接口统计信息、伪装链接和多播成员(multicast memberships),ss 命令用来显示网络套接字信息, 它允许显示类似 netstat 一样的信息。关于 ss 和 netstat 使用, 可参考下列资源。

相关链接:

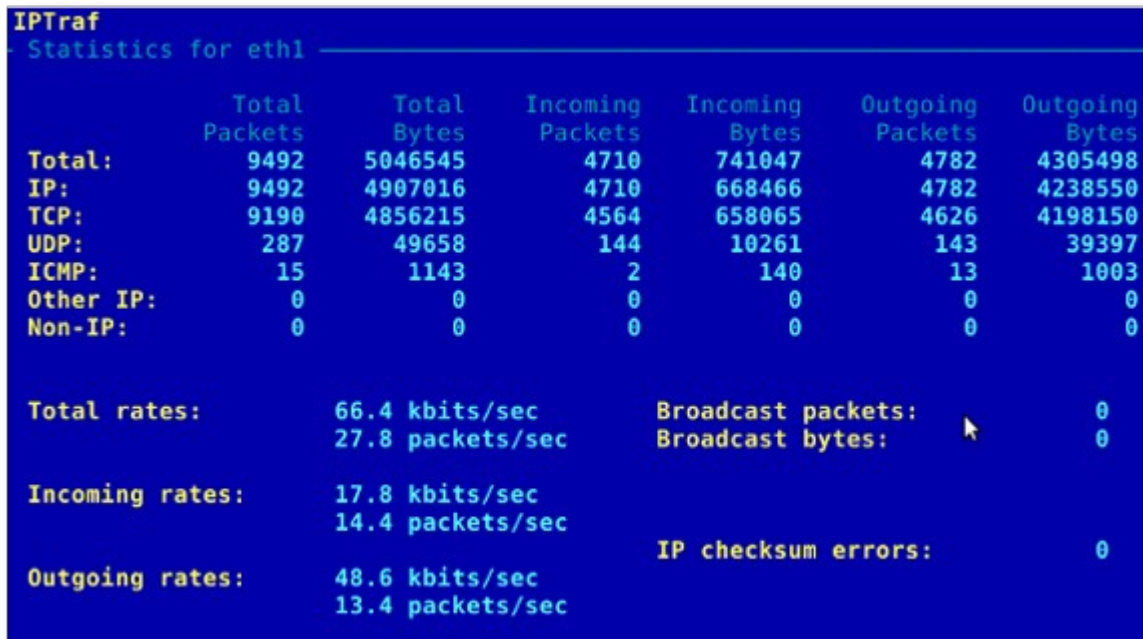
- [ss: Display Linux TCP / UDP Network and Socket Information](#)
- [Get Detailed Information About Particular IP address Connections Using netstat Command](#)

#13: iptraf - 网络实时信息

iptraf 是一个可交互式的 IP 网络监控工具。它可以生成多种网络统计信息包括: TCP 信息、UDP 数

量、ICMP 和 OSPF 信息、以太网负载信息、节点状态、IP 校验错误等。有下面几种信息格式：

- 不同网络 TCP 链接传输量
- 不同网络接口 IP 传输量
- 不同协议网络传输量
- 不同 TCP/UDP 端口和不同包大小网络传输量
- 不同第二层地址网络传输量



The screenshot shows the output of the IPTraf utility for the eth1 interface. It displays a table of statistics including total packets, bytes, incoming/outgoing packets, and bytes, categorized by protocol (Total, IP, TCP, UDP, ICMP, Other IP, Non-IP). It also shows rates in kbits/sec and packets/sec for total, incoming, and outgoing traffic, as well as broadcast packets/bytes and IP checksum errors.

	Total Packets	Total Bytes	Incoming Packets	Incoming Bytes	Outgoing Packets	Outgoing Bytes
Total:	9492	5046545	4710	741047	4782	4305498
IP:	9492	4907016	4710	668466	4782	4238550
TCP:	9190	4856215	4564	658065	4626	4198150
UDP:	287	49658	144	10261	143	39397
ICMP:	15	1143	2	140	13	1003
Other IP:	0	0	0	0	0	0
Non-IP:	0	0	0	0	0	0

Total rates:	66.4 kbits/sec	Broadcast packets:	0
	27.8 packets/sec	Broadcast bytes:	0
Incoming rates:	17.8 kbits/sec		
	14.4 packets/sec		
Outgoing rates:	48.6 kbits/sec	IP checksum errors:	0
	13.4 packets/sec		

图 02：一般接口信息：不同网络接口 IP 传输量

Src	Pst	Size	Win	Size	Flags	Iface
43.222.232.20:45359	74.86.48.99:80	52	65535	390	--A-	eth1
74.86.48.99:80	59.92.30.94:38616	570	137	31	-PA-	eth1
74.86.48.99:80	192.168.158.213:11690	587	31	31	-PA-	eth1
74.86.48.99:80	75.126.168.152:80	52	65535	39	--A-	eth1
157.127.124.15:62884	74.86.48.99:80	52	65535	40	CLOSED	eth1
74.86.48.99:80	41.219.249.101:1834	52	6432	49	-PA-	eth1
75.126.168.152:80	192.168.158.213:11690	52	65535	49	CLOSED	eth1
72.223.24.90:57889	74.86.48.99:80	52	65535	40	-PA-	eth1
74.86.48.99:80	213.47.93.188:50694	1420	20	20	--A-	eth1
75.126.168.152:80	74.86.47.178:40077	46	4135	48	CLOSED	eth1
74.86.48.99:80	74.86.48.99:80	52	272	27	CLOSED	eth1
76.79.231.30:39049	74.86.48.99:80	52	27	27	CLOSED	eth1
74.86.48.99:80	17.43.28.16:65186	0	0	0	----	eth1
74.86.48.99:80	74.86.48.99:80	52	41	41	--A-	eth1
17.43.28.16:65186	74.86.48.99:80	52	65535	27	-PA-	eth1
74.86.48.99:80	158.230.106.182:45944	855	27	27	-PA-	eth1
74.86.48.99:80	24.122.34.154:64223	46	65535	0	--A-	eth1
74.86.48.99:80	74.86.48.99:80	46	4280	39	CLOSED	eth1
74.86.48.99:80	68.177.214.106:48483	40	31	31	--A-	eth1
59.92.30.94:38615	74.86.48.99:80	586	110	0	----	eth1
17.43.28.16:65186		1492	31	31	-PA-	eth1
		499	65535	31	-PA-	eth1

图 03: 不同网络 TCP 链接传输量

#14: tcpdump: 详细的网络流量分析

tcpdump 是一个简单网络流量转储工具，然而要使用好需要对 TCP/IP 协议非常熟悉。例如要显示关于 DNS 的网络流量，输入：

```
# tcpdump -i eth1 'udp port 53'
```

显示所有进出 80 端口 IPv4 HTTP 包，也就是只打印包含数据的包。例如：SYN、FIN 包和 ACK-only 包输入：

```
# tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'
```

显示所有到的 FTP 会话，输入：

```
# tcpdump -i eth1 'dst 202.54.1.5 and (port 21 or 20)'
```

显示所有到 192.168.1.5 的 HTTP 会话

```
# tcpdump -ni eth0 'dst 192.168.1.5 and tcp and port http'
```

用 wireshark 浏览转储文件中的详细信息，输入：

```
# tcpdump -n -i eth1 -s 0 -w output.txt src or dst port 80
```

#15: strace - 系统调用

追踪系统调用和型号，这对于调试 Web 服务器和其他服务器非常有用。[了解怎样追踪进程和他功能。](#)

#16:/proc 文件系统 - 各种内核信息

/proc 目录下文件提供了很多不同硬件设备和内核的详细信息。更多详情参见 [Linux kernel /proc](#)。一般/proc 例如：

```
# cat /proc/cpuinfo
# cat /proc/meminfo
# cat /proc/zoneinfo
# cat /proc/mounts
```

#17:Nagios - 服务器及网络监控

[Nagios](#) 是一款非常流行的系统及网络监控软件。你可以轻松监控所有的主机、网络设备及服务。它能在发生故障和重新恢复后发送警讯。FAN 是 "Fully Automated Nagios" 的缩写。FAN 的目标就是由 Nagios 社群提供 Nagios 的安装。为了使安装 Nagios 服务器更加容易，FAN 提供一个标准 ISO 格式的光盘镜像。此发行版中还会包含一组增强用户使用体验的工具。

#18:Cacti - 基于 Web 的监控工具

Cacti 是一套完成的网络图形化解决方案，基于 RRDTool 的资料存储和图形化功能。Cacti 提供一个快速的轮询器、进阶的图形化模板、多种 数据采集方法和用户管理功能。这些功能都拥有非常友好易用的界面，确保可以部署在一个包含数百台设备的复杂网络中。它提供关于网络、CPU、内存、已登录用户、Apache、DNS 等信息。关于怎样在 CentOS / RHEL 安装配置 Cacti，详见：<http://www.cyberciti.biz/faq/fedora-rhel-install-cacti-monitoring-rrd-software/>

#19:KDE System Guard

KSysguard 是在 KDE 桌面下一个网络化的系统监控工具。这个工具可以通过 SSH 会话运行。它提供很多功能，例如可以监控本机和远程主机的客户端/服务器架构，前端图形界面使用所谓传感器得到信息并展现出来。传感器返回的可以是一个简单的数值或是一组表格的信息。针对不同的信息类型，提供一个或多个显示。这些显示被组织多个工作表中，可以工作表可以独体储存和加载。所以，KSysguard 不只是一个简单的任务管理器，还是一个可以控制多台服务器的强大工具。

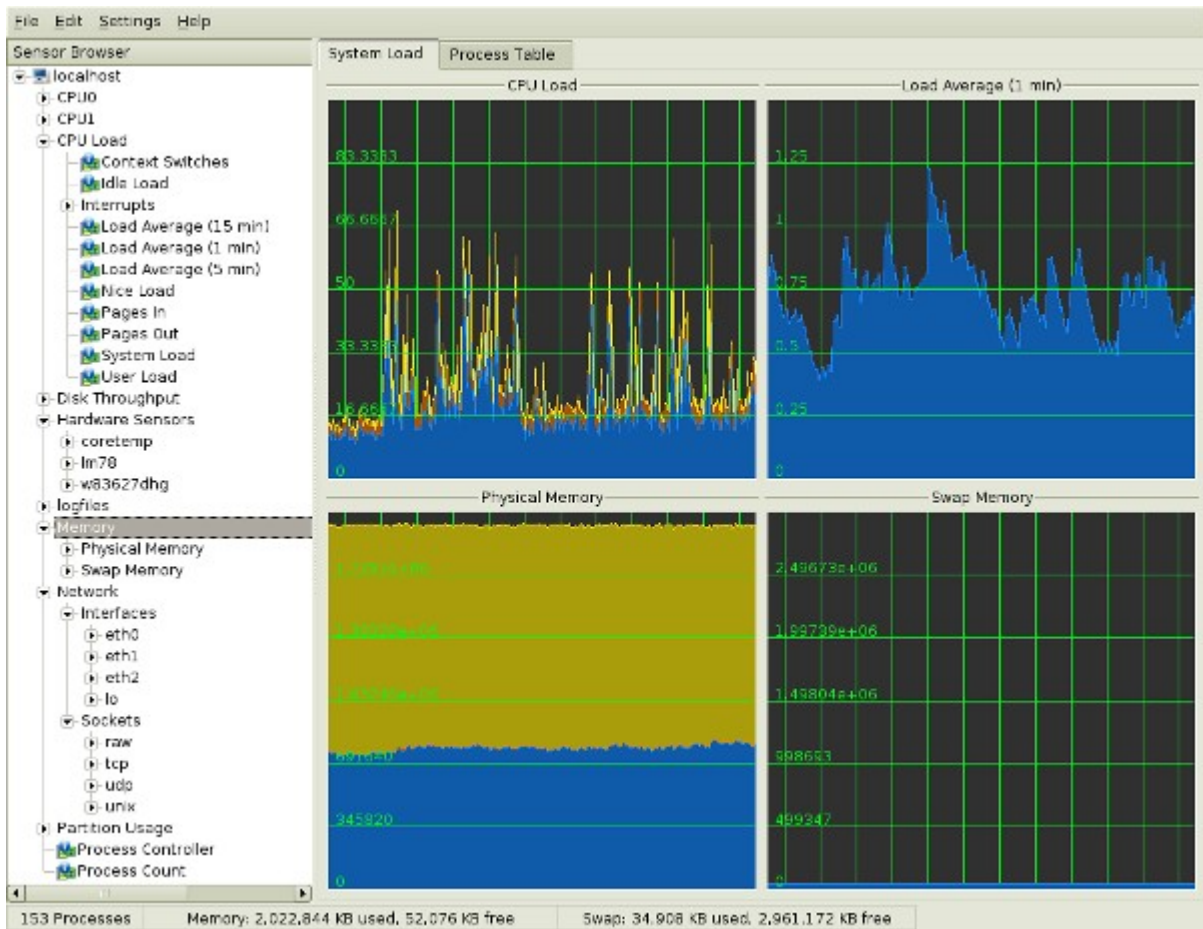


图 05: KDE System Guard

详细用法参见: [the KSysguard handbook](#)

#20:Gnome System Monitor

System Monitor 可以显示系统基本信息、监控系统进程、系统资源及文件系统使用率。你也可以使用 System Monitor 监控和修改系统行为。尽管没有 KDE System Guard 功能强大,但其提供的基本信息对于入门用户还是非常有用的。

- * 显示关于计算机硬件和软件的各种基本信息。
- * Linux 内核版本
- * GNOME 版本
- * 硬件
- * 安装的内存
- * 处理器及其速度
- * 系统状态
- * 当前可用的硬盘空间
- * 进程
- * 内存及交换空间
- * 网络使用率
- * 文件系统
- * 所有挂载的文件系统及其基本信息

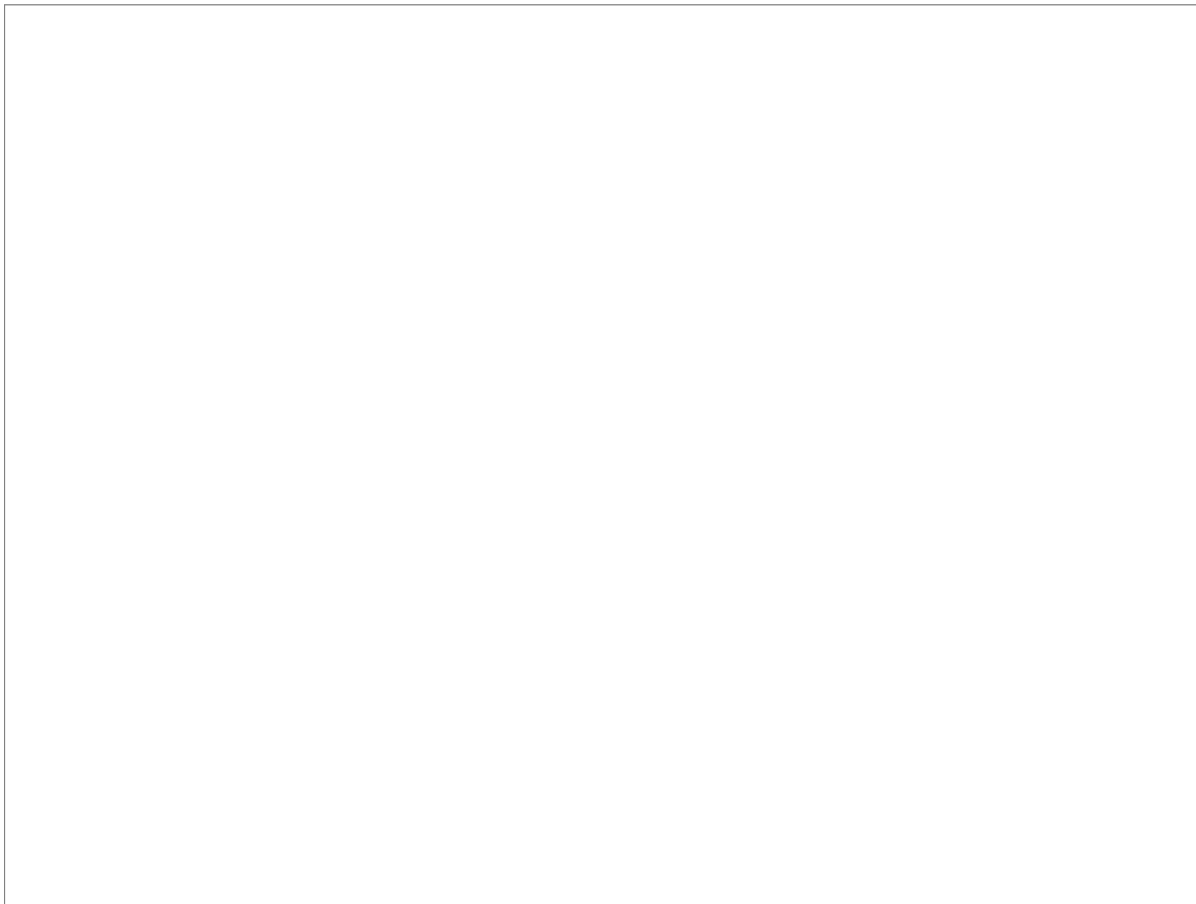


图 06: The Gnome System Monitor application

原文地址: <http://www.cyberciti.biz/tips/top-linux-monitoring-tools.html>

网友热评

热点技术评论

[JAVA 开发首选工具](#)

[搭建 ASE Cluster Edition 步骤](#)

[python 有处理 unicode 转义的函数吗？](#)

[FreeBSD 如何安装个桌面？](#)

[如何实现一个简单高效的 atoi](#)

[Python 能否实现服务器集中管理功能？](#)

[crontab 如何指定每小时执行任务](#)

[PostgreSQL 与 MySQL 比较](#)

[freebsd 每周这样维护够吗？](#)

[问一个有关 sed 匹配模式的话题](#)

[如何彻底删除 sybase 数据库表空间](#)

[为什么 C 语言效率比 C++ 高](#)

[Nginx 和 apache 哪个好？](#)

[web 服务器安装什么系统合适呢？](#)

[看 openvpn 源码困难](#)

[增量备份和差异备份到底有什么区别？](#)

[一次 Solaris 10 SMF 服务管理恢复案例](#)

[批量替换文件名字](#)

[SFW HA 是否会降低磁盘阵列的性能？](#)

[数据迁移问题，请大家给点思路](#)

[ZFS 基本命令](#)

[请教个手工安装网卡的问题](#)

[怎么测试内核的性能啊？](#)

[在内核中创建 HTTP Response 的疑惑](#)

[Framebuffer 原理、使用、测试系列文章](#)

热点新闻评论

[以后不能用 firefox 了](#)

[C++ 之父开发新语言 C+](#)

[来看看真正有技术含量的面试题！](#)

[APUE 到底怎么学习啊？](#)

[有感 redhat 和 microsoft 停止支持安腾](#)

[as400 下的程序如何导出](#)

[颤抖吧！专业程序员！](#)

[男子转嫁黑客攻击致金盾网瘫痪](#)

[你觉得 android 有前途不？](#)

[大胖球：微软之走向死亡](#)

[千万别学 python](#)

[开发推荐组合：vc6+winscp+putty](#)

[一个超复杂的 shell，想破脑袋了](#)

[Oracle 限制 Sun 主机支持服务](#)

[大家觉得面向对象怎么样](#)

[黑客称 Windows 比 Mac 安全](#)

[opensolaris 将死](#)

[程序员才看得懂的笑话](#)

[“龙芯 3A” 国产万亿次高性能计算机研制成功](#)

[看完几篇学术论文，发发牢骚](#)

[听说 yahoo 不用 bsd 了](#)

[CHINAUNIX 走过 10 年](#)

[下辈子，再也不做 c](#)

[面试归来，严重被打击](#)

[这就是 linux 的好处](#)