51CTO.com
技术成就梦想

Linux 运维趋势

第五期

2011年2月号 内网, 开发环境, 虚拟化, 版本控制, FreeBSD

内网开发环境

内容目录

Linux 运维趋势

杂志策划：51CTO 系统频道

本期主编：杨赛

封面制作：高鹏飞

交流圈子：
http://g.51cto.com/linuxops

专题页面邮件订阅入口：
http://os.51cto.com/art/201011/233915.htm

下载汇总页：
http://down.51cto.com/zt/71

投稿信箱：
yangsai@51cto.com

我不喜欢用“获利”这个词来形容这一过程，因为这个词暗示了一种便利，仿佛人们无需牺牲就可以享用。这并非是对自由的正确诠释。

Stallman 最新访谈：Debian，Mono 与智能手机的自由化

采访/Jacob Barkdull
编译整理/杨赛

人物简介：

Richard Matthew Stallman 是美国自由软件运动的精神领袖、GNU 计划以及自由软件基金会的创立者。作为一个著名的黑客，他的主要成就包括 Emacs 及后来的 GNU Emacs，GNU C 编译器及 GDB 调试器。他所写作的 GNU GPL 是世上最广为采用的自由软件许可证，为 copyleft 观念开拓出一条崭新的道路。

本期的访谈内容与系统运维并没有直接关系，与 Linux 的关系也不是很大。相对于开源运动带来的技术价值，自由软件运动更多是一种哲学。开源和自由软件是一种互相包括的关系——开源项目的创造方式决定了其对自由软件的促进，而对自由软件怀抱理想和憧憬的开发者们则通过参与开源项目而将理想付诸行动。本期的人物，Richard M. Stallman 是世界上最出色的开发者，黑客与哲学家之一。和 Stallman 口中的用户一样，系统管理员们也需要了解自由。虽然自由并非总是我们最需要的东西，但了解自由软件的思想无疑能让我们在很多方面走得更远。

本文的采访者 Jacob Barkdull 是一位致力于推广自由软件的音乐家，在技术、科学、政治领域发表文章，熟悉有关 Unix，GNU 和 Linux 的各方面背景。

Jacob：我们常常听到自由软件怎样让社会获利，那么你自己呢？你自己是如何通过自由软件获利的？

Stallman：自由软件运动让我从专有软件中逃离出来。我现在正在一个上网本上进行回复，这上网本上所有的软件都是自由的：从 BIOS 开始往上的所有软件。我并不是那些把用户当做白痴的软件们的一个无助的客户，而是自由社区的一员。

自由软件运动的发起正是为了让所有人享受到这种自由。目前为止，还不是所有人都享受到了这种自由，但像我们这样愿意做出一些牺牲的人们都能够得到这种自由。通常，我不喜欢用“获利”这个词来形容这一过程，因为

这个词暗示了一种便利，仿佛人们无需牺牲就可以享用。这并非是对自由的正确诠释。

Jacob：目前你最为之兴奋的软件/项目/运动有哪些？

Stallman：我认为 Diaspora 将变得十分重要。（译注：Diaspora 是一个新兴的内容共享服务。）另外，我们也需要让智能手机可以完全运行在自由软件之上。

Jacob：我们看到 Debian Squeeze 已经在 Linux 内核包中将自由固件与非自由固件隔离，而且可能将在未来的默认软件包中完全排除非自由固件。你认为 Debian 正在向一个完全自由的发行版发展么？现在使用 Debian 是否有什么不好的地方？

Stallman：要成为一个自由的发行版，意味着 Debian 必须在它的自由软件包和服务器上将所有对非自由软件和代码的引用去除。（很多贡献到 Debian 社区的软件包都是为了在 Debian 上安装来自其他开发者的非自由软件的。）

而根据我目前的了解，Debian 还没有这个意向。我希望他们考虑往这个方向发展。

Jacob：在很多人看来，Mono 的专利纠纷可能会是 GNU/Linux 和自由软件的一个大问题。对于使用 Mono 进行自由软件的开发和贡献，你认为长远而言对用户的自由是好事还是坏事？相比使用专有软件而言。

Stallman：这是一个种苹果和吃桔子之间的比较。与其尝试解答这个含糊不清的问题，我认为我应该将 C#和 Mono 的问题解释清楚。

Mono 是 C#的自由实现。你没法儿“用 Mono 编写程序”；你使用 C#编写程序。如果你用 C#写一个程序，那么用 Mono（自由软件）好过用微软.NET（专有软件）。

如果你要编写程序，我建议你使用 C#之外的语言。如果你在自由领域使用 C#，你可能会遭到微软专利的恐吓。所以不用 C#的话，就避免了这个问题。无论你的软件是自由软件还是只是个人使用的私人软件，上面这一条都是个好建议。（当然如果你打算发布一个专有软件，那么无论你使用何种语言，都应该拒绝参与。）

Jacob：为每一个专有的 web 技术实施一个自由软件的替代品——你认为现在到时候了么？

Stallman：是的——而且这并不仅仅意味着我们做一个自由软件出来，这个软件在服务器上和专有软件做一样的工作。我们需要编写那些能够在用户自己的机器上运行的自由软件或者能够以 P2P 方式运行的自由软件。这样才能让用户拥有完全的控制权。

Jacob：那些手机巨头们正在往自由软件的方向一点点靠拢，对此你是怎么看的？你认为这行动是原则上的，还是仅仅作为一种实用的手段？

Stallman：这些公司对我们的自由没什么兴趣，你可以从他们的行为中看出这一点。

比如说，Android 的源代码是自由软件，但是很多手机的可执行文件是非自由的，因此用户无法在手机上运行他们自己的版本。还有很多非自由的固件，非自由的应用。是的，这让我们离没有专有软件的手机更近了一步，但目标还十分遥远。

还有很多工作要做。

Jacob：如果 Linux 基金会一直按照自由软件基金会的尺度来推动软件的自由化，将“Linux”做为一个操作系统的名称还会造成今天这样的问题么？（译注：理论上，Linus 开发的 Linux 是操作系统的内核部分，而大多数 Linux 发行版其实是 GNU 操作系统和 Linux 内核加上其他软件打包而成的，而且 GNU 的代码在系统中的比重是最高的，所以 Stallman 一直坚持“Linux 系统”应该被称呼为“GNU/Linux 系统”。）

Stallman：嗯，那样的话问题就小多了。将一个 GNU 系统称呼为“Linux”对 GNU 项目而言仍然不太公平，但是公平的重要性没有用户自由的重要性那样高。

而现在，你对这个系统的称呼对两方面都有影响。（译注：即称呼“Linux”或“GNU/Linux”在公平性和用户自由方面的含义都是不同的。）Torvalds 并不认为用户的自由是最终目标，而当人们将 GNU/Linux 的开发都算在他的名下时，他通过他的影响力降低了人们对自由的追求。所以如果大家更多的提及 GNU 项目，这会帮助我们建立可以与之平衡的影响力。

Jacob：即使在今天，人们听到“自由软件（Free Software）”的第一反应仍然是“免费软件”。你认为这在未来会改变么？或者我们需要为“自由软件”找一个更准确的词？

Stallman：我不会预知未来。但是大多数人在听过“是自由言论，不是免费啤酒（free speech, not free beer）”这样的说明之后，都能够明白我们的意思。

我们无法使用“freedom software”这个词，因为这个名字被一个公司占用了。不过我们可以用很多方法来更准确的表达我们的意思比如，你可以说“free/libre software”或“libre software”。

Jacob：有些人抱怨“bug 回归”的问题。你认为是什么原因导致自由软件中出现这种问题？

Stallman：我没见到这种问题，我对这种说法表示怀疑。

Jacob：你平时最喜欢看哪些网站？当然，除了 fsf.org 和 gnu.org 之外。

Stallman：通常，我查看网页的方式是往一个地址发送一个消息，通过运行脚本将网页内容邮件给我。这种方式主要用于查看别人推荐给我的文章，而不是用于浏览。

我基本只在别人的机器上偶尔浏览网页，因为我没精力定期的浏览某一个站点。

即使我可以每天浏览某个站点，我也没有足够的时间。我必须完成的工作占用了几乎所有的时间。所以我的做法是，邀请一些志愿者定期的浏览各个站点，并将我可能感兴趣的文章邮件发给我。▌※

本专访为节选，完整内容请参考英文原文：
Interview with Richard Stallman (2011)

译文全文：
http://os.51cto.com/art/201102/244556.htm

推荐阅读：

自由软件赚钱史回顾

隐逸在 GNU/Linux 中的非自由软件

斯托尔曼：云计算概念愚蠢之极

过去的主要问题是磁盘带宽。Mollom 需要跟踪互联网上所有 IP 地址和所有 URL 的信誉，所以这是一个随机访问频繁的庞大数据存储区。

基于 Cassandra 和 Glassfish 的高效垃圾信息过滤系统

文/Todd Hoff
编译/布加迪

Mollom 是一个新兴的垃圾信息过滤系统，自从其打开数字检查系统以来，它已经拒绝了 3730 万条以上的垃圾信息。

Mollom 建立了一套非常高效的学习系统。为了一探究竟，我采访了 Mollom 的创办人之一 Benjamin Schrauwen 以及 Glassfish 和 Java Enterprise 方面的专家 Johan Vos。

统计数字

◆Mollom 服务于 40000 个活跃网站，其中许多是大客户，比如索尼音乐公司、华纳兄弟福克斯新闻网和《经济学家》杂志社。

◆每天查出 50 万条垃圾信息。

◆每秒处理 100 个 API 调用。

◆垃圾信息检查方面的延迟用时在 30 至 50 毫秒。最慢的连接用时 500 毫秒。5%以下的延迟是 250 毫秒。

平台

◆两台生产服务器在两个不同的数据中心运行，确保故障切换。

◆一台服务器放在美国东海岸，另一台放在西海岸。

◆每台服务器配备英特尔至强四核 2.8GHz 处理器、16GB 内存和 4 只采用 RAID 10 配置的 300 GB 硬盘。

◆SoftLayer–机器由 SoftLayer 托管。

◆Cassandra–选择了 NoSQL 数据库，原因是写入性能高，还能够跨多个数据中心来操作。

◆Glassfish–面向 Java EE 平台的开源应用服务器。选择 Glassfish 的理由是，它具有企业就绪功能，比如复制和故障切换。

◆Hudson–提供了跨所有服务器，不断测试和部署后端系统的功能。

◆Java–Mollom 一开始就是用 Java 编写而成的。

◆Munin–用于测量和描绘关于服务器运行状况的度量标准。

◆MySQL–JPA 用于普通的数据集，Cassandra 则用于庞大的数据集。

◆Pingdom–用于监测正常运行时间。

◆Zendesk–用于支持。

◆Drupal–用于使用自定义电子商务模块的主网站。

◆Unfuddle–分布式开发团队使用 Subversion 控制源代码。

Mollom 是什么？

Mollom 是一项 Web 服务，用于将各种类型的垃圾信息从用户生成的内容中过滤掉，这些内容包括：评论留言、论坛帖子、博客帖子、民意调查、联系表单、登记表单和密码请求表单。确定垃圾信息的依据不仅仅有所发的内容还有发帖者过去的行为和信誉。

Mollom 是怎么使用的？

Drupal 等应用程序使用模块（Module）来整合 Mollom；模块可以将自己安装到内容编辑集成点上，可以在内容写到数据库之前检查垃圾信息。这个过程就像这样：

◆用户将评论提交到网站上后，对后端服务器进行 API 调用。

◆内容经过分析；如果是垃圾信息，就会告知网站阻止内容；或者如果后端服务器不确信它会建议网站显示验证码，后端服务器同时会提供验证码。

◆验证码正确填写后，内容将得到接受。在大多数情况下，人是看不到验证码的，内容将直接作为非垃圾信息的正常信息（ham）而被接受。

操作过程

◆安装。对于 Drupal 来说，安装很容易。可以把它当作其他任何模块那样来安装。先在 Mollom 网站上设一个帐户。获得一对安全密钥，把这些密钥配置到该模块中，然后选择你想用 Mollum 来保护系统中的哪些部分。

◆日常操作。定期检查，看看垃圾信息有没有进入。要是有垃圾信息进入，就要告诉 Mollom。这个过程其实在帮助训练 Mollom 的机器学习算法，明白什么是垃圾信息。

◆允许匿名用户交互。一旦你需要注册，用户的参与热情就会大大减低，而注册阻止不了垃圾信息发送者。

架构

◆每台服务器都能处理所有请求，但它们都有完整的故障切换机制。工作负载在多台机器之间分配。如果一台机器出了故障，那么工作负载会转移到另一台机器上。

◆Mollom 过去有三台服务器，但由于大大提升了性能，所以可以将第三台服务器用作登台服务器（staging server）。

◆每台服务器每秒能够处理整整 100 个连接，每个连接执行整个流程：全文本分析、计算作者的信誉以及提供验证码。

◆真正为低延迟进行了优化。由于垃圾信息检测是内容提交到网站的整个过程的一部分，要是检测所花时间很长，用户会觉得很烦人。

Mollom 经历了几个发展阶段：

1、最初，只有两个人的小团队业余时间开发算法、分类器以及 Mollom 要解决的真正的商业问题。为了扩建后端系统上的基础架构，他们实施了自己开发的线程池、连接池和资源管理机制。这费时又费力。后来他们改用 Glassfish，就不用太担心内存管理、REST 处理、XML 解析和数据库连接池。

2、过去的主要问题是磁盘带宽。Mollom 需要跟踪互联网上所有 IP 地址和所有 URL 的信誉，所以这是一个随机访问频繁的庞大数据存储区。

3、在早期，Mollom 使用廉价的虚拟机，一切都放在无法进行扩展的 MySQL 中。

4、随后，Mollom 又改用固态磁盘，把一切存储在文件中。固态磁盘解决了写入问题，但存在这些问题：

◆价格很昂贵。

◆对于所安装的文件系统的类型非常敏感。

◆写入速度快，但是对数据进行迭代处理，清理数据，以及通过成千上万个小对象来训练新的分类器等过程的速度还是相当慢。

5、随后，Mollom 由固态磁盘改用 Cassandra。

◆Cassandra 先用作写入密集型工作负载的数据库，并用作一个缓存层：

◆运行在 RAID 10 磁盘配置上（经条带化和镜像处理），非常适合于密集的读/写操作。

◆Cassandra 针对写操作进行了优化，Mollom 面临的写操作比读操作多得多。

◆被设计成可以在数据中心里面分布，也可以跨数据中心分布。

◆一个缺点是没有标准的 NoSQL 接口，因而很难编写应用程序。

◆Cassandra 的行缓存（row caching）让 Mollom 不必往系统中添加另一个缓存层，这消除了好多的应用程序代码。

◆Cassandra 拥有老化功能：经过一段时间后，会自动删除数据。欧洲订有严格的隐私法，要求某些数据在一段时间后必须予以清除这项功能因而大受欢迎。

博客评论在整个系统中的路径如下：

◆请求可能来自任何客户端。客户端跨服务器对请求进行负载均衡处理。典型的客户端是 Drupal 系统。请求可能是 XML-RPC 请求或 REST 请求。

◆请求由 GlassFish 处理，遵循典型的应用服务器工作流程：请求由服务器小程序来处理，并委托给会话组件（session bean）。

◆Mollom 有不同的功能部分：垃圾信息检查和验证码处理。验证码包括：生成、提供和处理响应。不同的会话组件负责 Mollom 功能的不同部分。

◆分类器完全在内存中。一小块内容被拆开来，分成了成千上万个可识别垃圾信息的小标记（token）。这些分类器在内存中大约有几百万个标记。分类器需要高速运行，它们必须在内存中。

◆Cassandra 里面保存的是信誉分数、频率、URL 和 IP 地址。Cassandra 新的行缓存功能现在充当其缓存层。以前 Mollom 实施了内部缓存，但现在内存缓存被拿掉了。

◆内存中的数据结构并不直接复制。它们写入到 Cassandra，再由 Cassandra 来复制。另一端的缓存有超时机制，所以它会访问 Cassandra，获取新的数据。内容最终是一致的。有一小段时间会出现不一致，但是在这么短的时间里，数据模型并未受到负面影响。

◆视具体情况来管理一致性。对于信誉和 IP 地址来说，最终一致性很好。包括验证码会话的会话数据严格做到完全一致，那样机器进行故障切换时，数据会正确迁移过去。

客户端负载均衡

◆Mollom 使用的客户端负载均衡基于延迟等标准来分配负载。作为一家新兴公司，Mollom 没有钱来买大型负载均衡工具。

◆客户列表通过 API 来进行管理。每个客户都有一份服务器列表，列出了可以使用的服务器。

◆每个客户可以使用不同的服务器，以缩短延迟。

◆当一台服务器发生故障时，客户会尝试连接列表中的下一台服务器。

◆客户端负载均衡有助于从旧系统迁移到基于 Glassfish 的新系统。

◆客户端方法的一个缺点是，要是第三方客户软件编得很差，就会有问题。比如说，客户获得服务器列表后，可能以相反的顺序对列表进行迭代处理，这其实是不对的。现在 Mollom 与客户的开发人员紧密合作，提供优质的参考代码示例。

发展过程

◆Mollom 是分布式团队。三个人在比利时另几个人在得克萨斯州、波士顿和德国。

◆Scrum。

◆开发人员在本地开发，然后将代码提交到 Unfuddle。

◆Hudson 作为 Mollom 的持续集成环境 Hudson 让 Mollom 更容易从旧的 Glassfish 系统迁移到新的 Glassfish 系统，因为只有先通过测试，才可以部署。

◆单元测试、系统测试和 Drupal 测试。只有 Hudson 通过这些测试，才可以部署系统。

◆部署仍是手动进行，以缩短潜在的停机时间。

◆每当 Mollom 发现垃圾回收时间存在问题总是归因于应用程序存在内存泄漏。若出现内存泄漏的情况，Mollom 就会分析核心转储（core dump）文件。Mollom 在亚马逊上租用一个庞大的内存实例来分析转储文件。处理堆转储文件大约需要 2 个小时。Mollom 比较两个转储文件，一个是执行 10 小时后的，另一个是执行 20 个小时后的。要是两者存在重大区别，那么很可能是由于内存泄漏。

未来方向

◆Mollom 将尝试使用 Glassfish 用于每个数据中心的负载均衡。

◆如果负载加大 10 倍，Mollom 将不得不添加更多的 Cassandra 节点。磁盘输入/输出是瓶颈。只有当发展需求增加 10 倍以上，才需要添加更多的应用服务器。

◆让堆栈来处理繁重任务，以减少代码。在早期，Mollom 代码库要比现在大得多。Cassandra 可以处理复制和行缓存；Glassfish 可以处理集群。代码库慢慢简化下来。

◆尽量减少有锁争用，才有可能保持真正的并行处理。▌※

本文为节选，完整内容请参考英文原文：
Mollom Architecture

译文全文：
http://os.51cto.com/art/201102/244509.htm

2011年1月到2月之间，发生了下面这些事……

八卦，趣闻与数字 2011.01 – 2011.02

收集整理/51CTO 系统频道

【RHEL】红帽在1月13日发布了其企业级Linux 5平台系列的最新版本，RHEL 5.6。支持更多的新硬件，虚拟化更新，DNS packages，完全支持Ext4文件系统。

http://os.51cto.com/art/201101/243070.htm

【openSUSE】自2011年1月13日起，openSUSE将停止向openSUSE 11.1提供安全或重要的修复以及软件更新。

http://os.51cto.com/art/201101/243754.htm

【Ubuntu】根据2月11号的消息，Ubuntu 11.04 Natty发布计划有所改变，将不会发布RC候选版，发布2个Beta版之后直接发布正式版。

http://os.51cto.com/art/201102/244402.htm

【Debian】Debian 6.0（代号Squeeze)于2011年2月6日正式发布，包含了KDE Plasma桌面和应用程序，GNOME桌面，Xfce以及LXDE桌面环境，还有所有的服务器应用程序。

http://os.51cto.com/art/201102/244388.htm

【Ubuntu & Qt】Mark：可能会在Natty之后的Ubuntu 11.10中内置Qt库及运行环境，同时会对一些有价值的Qt程序进行评估，并考虑会把它们内置于安装光盘及默认安装到Ubuntu中。

http://os.51cto.com/art/201101/243741.htm

【Qomo】Qomo 1.2正式发布，在Qomo 1.1的基础上修正了一些Bug，修正了引导脚本，加入了支持winxp和win7硬盘安装的工具qomowin，KDE版本也成功更新到4.5.5。

http://os.51cto.com/art/201101/243746.htm

【LibreOffice】OpenOffice分支LibreOffice办公套件于2011年1月25日正式宣布发布首个稳定版——LibreOffice 3.3 Final。

http://os.51cto.com/art/201101/244055.htm

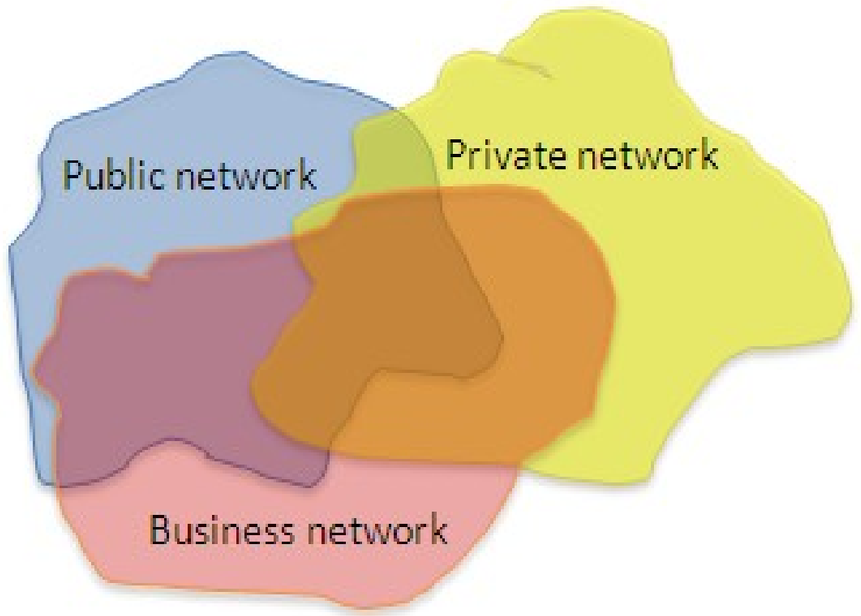
【Oracle Linux】Oracle Linux 6发布，主要的改进包括：Ext4文件系统、XFS、ftrace、Linux性能计数器和perf、Powertop、Latencytop。

http://os.51cto.com/art/201102/244385.htm

【人机大战】参与了本届大型Jeopardy!比赛的IBM Watson超级计算机运行在SUSE Enterprise Linux Server 11和IBM Power 750的10架服务器上。

http://os.51cto.com/art/201102/244653.htm

本期专题：内网开发环境



开发环境其实就是网站成熟代码的前身。只有保证这个工作做得稳定了，你的网站才能够稳定。

本期专题主要内容由抚琴煮酒提供，重点关注了基于 FreeBSD+PHP 的内网开发环境。

我也尝试过一段时间 Git，但由于我们的项目管理本来就是基于集中式管理，所以 git 目前暂时不是太适合公司。

基于 FreeBSD+PHP 企业内网开发环境的基本思路

文/抚琴煮酒

对于企业内网开发环境，很多系统管理员容易忽略或干脆忽视这个问题其实这是一个错误的做法，这牵涉到二个问题：

一、开发环境其实就是网站成熟代码的前身，只要保证这个工作做得稳定了，你的网站才能够稳定；

二、如果你的开发团队是一个 500 人以上的大团队，你就会明白这个专题的作用，因为牵涉到许多诸如代码安全及管理的问题。

下面我将从七个方面介绍企业内网开发环境部署和管理方面需要考虑的问题。

一、系统的选择问题

抚琴煮酒建议的主要是二种选择：第一个就是 CentOS，第二是 FreeBSD。我们的单位最后选择将 FreeBSD 作为开发环境的原因其实也简单，就是它的稳定和方便性。稳定性这个是大家公认的，而 FreeBSD 部署环境确实是方便和快速的，因为开发无论怎样，环境总是第一的，这个有需要验证的朋友可对比下在 CentOS 下源码部署 nginx 环境和 FreeBSD 部署 nginx，大家也可以区分下。操作系统我们选择的都是 64 位的，先前一批安装的机器均是 64-bit 的 FreeBSD 8.0，最近新装的均是 64bit 的 FreeBSD 8.1，没办法，虽然 FreeBSD 可以直接通过命令生级，但都是有代码的机器，万一出问题会严重影响整个部门同事的工作，想了很久就暂时共存吧，毕竟稳定第一。

二、服务器的选择

对比了当时一些市面上比较流行的服务器，我们最终选择了 Dell PowerEdge R710，Dell PowerEdge R710 作为 DELL PowerEdge 2950 的全面升级产品，无论从性能上还是管理功能上都得到了质的提升。在实际测试中表现令人惊艳，多任务处理中表现出强大的性能，在节能方面同样表现优异，并且噪音控制相对于上一代产品也得到了很大的改善。戴尔的新一代 OpenManage 功能丰富，设置服务器温度、功耗阀值和即时报警等独具特色的功能都让我们觉得此款服务器的性价比绝对是最优的，硬盘做的是 RAID5，其它方面也没什么好说的。

三、虚拟化软件的选择

因为选择的是 FreeBSD，在这个平台上也没什么好虚拟软件可选择；我们就用的是 FreeBSD 8.1 自带的 jail，通过长期的使用

和观察，确实在稳定性和开发上没什么问题，除了性能上跟专业的商业虚拟软件有差距外；前期的一批机器已将其用于了线上环境，目前发现非常稳定。

四、版本控制工具相关

公司成立的比较早，最先一批程序员都是用 CVS 版本控制工具来管理他们的代码；后期，我们将自己的 CVS 服务器生级成了 SVN，但总部那边也保留了 CVS 服务器，所以这二种服务在公司内是共存的；我也尝试过一段时间 Git，但由于我们的项目管理本来就是基于集中式管理，所以 git 目前暂时不是太适合公司。

五、文件服务器的选择

由于我们办公环境清一色是 windows xp 和 windows 7，这样就存在一些 Linux 和 windows 共享文件的问题。我们许多同事(包括我)都非常喜欢 TortoiseSVN 和 TortoiseCVS，本着安全的原则，我特地拿了一台机器专门作 checkout 机器，并且在上面配置了 samba 服务，并且作了相当严格的权限控制，方便大家在 windows 下用"映射网络驱动器"来进行代码的查看工作；此外，一些不是代码的文件(比如软件)，也用 samba 作了文件服务器，方例大家交流共享使用。

六、FTP 服务器的选择

我以前花了大量精力和时间测试，发现稳定和安全方面，vsftpd 确实当之无愧；另外，我也测试过 vsftpd 和 pureftpd，发现在功能上 pureftpd 确实强大和完美，但简便性上就完全输给了 vsftpd；vsftpd 服务我们主要是用于作数据库及 SVN、CVS 的备份，我们的备份原则是通过 shell 脚本，本机备份一次后再通过 vsftpd 再备份一次，有备无患，免得发生硬盘损毁的杯具事件。

七、WEB 开发环境的选择

我个人的职业之一是项目实施工程师，说老实话，用源码安装 LAMP 和 LNMP 环境确实是一件漫长和复杂的过程，在开发环境下，配置 FAMP 确实是一件易事，这个有时间和兴趣的朋友也可测试下。

在内网开发环境的整体部署过程，我们尝试将 SVN、CVS、samba 及 Jail 这些服务糅合在一起，目前发现效果总体来说还是不错的如果你是系统管理员，也不妨换个思路和做法尝试下这些你可能没有试过的服务，更高效和方便的部署你的开发环境。▌※

原文：
http://os.51cto.com/art/201101/242688.htm

推荐专题资源：

企业内网开发环境部署与管理全攻略

系统运维秘诀大分享

FreeBSD 入门指南

推荐文章：

FreeBSD 8 下如何最有效率的安装软件

安装 FreeBSD8 服务器后建议做的事

Linux 系统管理员都应该熟悉的工具

FreeBSD+Nginx+PHP 配置高性能 Web 平台

Jail以多种方法改进了传统的chroot环境概念。

FreeBSD 8.0下jail虚拟机完全实践

文/抚琴煮酒

在Linux下实现虚拟化，我们使用Xen或者KVM；在Windows下实现虚拟化，我们使用VMware或Hyper-V。相对应的，说到BSD系统下的虚拟化解决方案，那么jail是不能不提的。严格来说jail算是一种安全工具，但经过了多年的开发，jail已经是一个相当成熟的虚拟化解决方案，十分适合布置小规模的开发服务器环境。本文将介绍FreeBSD 8.0下使用jail来搭建一个小规模开发环境的详细步骤。

jail是什么

BSD类操作系统从BSD4.2开始提供了chroot。 chroot工具能够改变一组进程的根目录的位置，从而建立一个与系统中其他部分相隔离的安全环境，在chroot环境中的进程将无法访问其外的文件或其他资源。正是由于这种能力，即使攻击者攻破了某一个运行于chroot环境的服务，也不能攻破整个系统。chroot对于那些不需要很多灵活性或复杂的高级功能的简单应用而言相当好用。另外，在引入chroot概念的过程中，曾经发现过许多跳出chroot环境的办法。尽管这些问题在较新的FreeBSD版本中已经修正，但很明显地chroot并不是一些用于加固服务器安全的理想解决方案。因此，必须实现一个新的子系统来解决这些问题，jail便应运而生了。

jail以多种方法改进了传统的chroot环境概念。传统的chroot环境只限制了进程能够访问文件系统的哪些部分，其他部分的系统资源（例如系统用户、正在运行的进程， 以及网络子系统）是由chroot 进程与宿主系统中的其他进程共享的。jail扩展了这个模型，它不仅将文件系统的访问虚拟化，而且还将用户、FreeBSD 的网络子系统，以及一些其他系统资源虚拟化。

jail应用环境

我们实际的开发环境用了一台8核CPU、16G内存的服务器充当宿主机，开了大约六七台jail机作内部开发和测试使用，效果还是比较让大家满意的。就是FreeBSD自身一样，简单和稳定是其主要特点；特别是相对于Vmware的ESXI而言，配置起来要简单很多。

安装前的准备工作：

宿主机的性能尽量高些，内存越大越好；

/usr目录越大越好，我分的/usr大约300-400G；

为了权限和安装的便利，我的操作均是以root进行。

虚拟机上 jail 的 IP 跟我的宿主 IP 分别为 192.168.43.128 和 192.168.43.129，物理 bridge 直接。

jail 的安装

①第一步就是为 jail 选择一个位置。

这个路径是在宿主系统中 jail 的物理位置。一种常用的选择是

```
/usr/jail/jailname
```

此处 jailname 是 jail 的主机名。对于“完整” 的 jail 而言，它通常包含了 FreeBSD 默认安装的基本系统中每个文件的副本。这里我创建了 apache 目的是做一个 apache 服务的 jail

```
mkdir -p /usr/jail/apache
```

②编译源码

```
cd /usr/src
```

/usr/src 可以选择用 sysisntall --> configure --> src --> DVD/CD 来安装，这样速度最快。

```
make buildworld
```

③新建 world

```
make installworld
 DESTDIR=/usr/jail/apache
```

④安装配置文件

```
make distribution
 DESTDIR=/usr/jail/apache
```

distribution 这个 make target 将安装全部配置文件，或者换句话说，就是将 /usr/src/etc/ 复制到 jail 环境中的/etc

※安装 jail 的过程也是熟悉 FreeBSD 目录结构的过程。

⑤安装 devfs

在 jail 中不是必须要挂接 devfs(8) 文件系统。而另一方面，几乎所有的应用程序都会需要访问至少一个设备，这主要取决于应用程序的性质和目的。控制 jail 中能够访问的设备非常重要，因为不正确的配置， 很可能允许攻击者在 jail 中进行一些恶意的操作。通过 devfs(8) 实施的控制，可以通过由联机手册 devfs(8) 和

devfs.conf(5) 介绍的规则集配置来实现，但为了以后能方便的 ssh 到 jail 上，这里建议安装。

我直接在宿主机的/etc/rc.conf 里添加如下内容

```
jail_apache_devfs_enable="YES"
# 在 jail 中挂接 devfs
```

⑥配置宿主机的 /etc/rc.conf，vim /etc/rc.conf，添加内容如下：

```
jail_enable="YES"
jail_list="apache"
jail_apache_rootdir="/usr/jail/apache"
jail_apache_hostname="你的主机名，如
 www.51cto.com"
jail_apache_ip="192.168.43.129"
jail_apache_exec="/bin/sh /etc/rc"
jail_apache_devfs_enable="YES"
```

/etc/rc.conf 里原有内容如下：

```
ifconfig_le0="inet 192.168.43.128
 netmask 255.255.255.0"
defaultrouter="192.168.43.2"
hostname="mail.ewiz.com"
ifconfig_le0_alias0="inet
 192.168.43.129 netmask 255.255.255.0"
#最后一行的目的是为了替 Apache 的 jail 添加
 IP
```

⑦使用 sh /etc/rc 来使 jail 生效，不需要重启，只需要 sh /etc/rc 以后，也就是使他 jail enable，就可以启用 jail 了，

然后就可以使用 jls 看到 jail 的状态了；当然也可以用/etc/rc.d/jail 脚本也可以用于手工启动或停止 rc.conf 中配置的 jail，如 /etc/rc.d/jail start apache。

jail 的管理及其配置

管理 jail 选择工具 jailexec，可先用 jls 找出运行 apache 的 jid

例如 jexec 1 passwd root 可以改变 jail 的 root 密码

jexec 1 csh 可以用 root 和/bin/csh 进入 jail 系统

如何启动名为 apache 的 jail 的 ssh

```
jexec 1 vi /etc/rc.conf
rpc_bind_enable="NO"
#以上为了在 rc.conf 中去掉网卡地址绑定
sshd_enable="YES"
hostname="你的主机名，如 www.51cto.com"
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
#以上用于关闭邮件干扰
ntpd_enable="YES"
ntpd_sync_on_start="YES"
#以上用于同步时间
named_enable="YES"
```

然后是重要操作 jexec 1 sh /etc/rc 重启虚拟机的服务，不然启动不了 jail 之 apache 的 ssh 的。

另外，在宿主机上建一个用于 apache_jail 的 ssh 用户

```
jexec 1 pw useradd admin && jexec 1
 passwd admin
```

记得把 yjwan 放入 wheel 便于进去可以 su root 账户

```
jexec 1 pw groupmod wheel -m admin
```

如何允许你的 jail 能够 ping

没有配置前你会很郁闷的发现，无论你 ping 啥，就会出现

```
ping: socket: Operation not permitted
```

如果想永久保留配置，可以在宿主机上面修改/etc/sysctl.conf 文件

加上

```
security.jail.allow_raw_sockets=1
```

重启 jail 虚拟机

```
/etc/rc.d/jail restart apache
```

jail 的优化

从宿主机 将/etc/resolv.conf 文件 复制到 jail 系统中

```
cp /etc/resolv.conf
 /usr/jail/apache/etc/resolv.conf
```

将宿主机的 make.conf 也复制过去，这样 ports 安装速度很快

```
cp /etc/make.conf
 /usr/jail/apache/etc/make.conf
```

在宿主机上将/usr/ports 挂接到 jail 上，此行可添加到/etc/rc.conf 上。

```
mount_nullfs /usr/ports
 /usr/jail/cas/usr/ports
```

※这里值得一说的是，由于宿主机有二个 IP：

192.168.43.128

192.168.43.129

所以我们 ssh jail 的时候，很有可能还是在宿主机上；只有 jail 的 ssh 配置成功才可能 ssh 上去，ssh 上去看，注意看下当前的 hostname 和 ifconfig。▌※

原文：
http://os.51cto.com/art/201009/224311.htm

用户的最终权限分配为目录的权限和 samba 权限的最小交集。

Samba 在企业中的简单应用案例

文/抚琴煮酒

本文介绍的 Samba 应用案例在很多公司需要，只要根据具体情况修改配置文件就可以了。其实原理也很简单，就是用户的读与写权限要分开。

案例要求与分析

需求 1

所有与员工都能在公司内漫游办公，但不管在那台电脑上工作，都要把自己的文件数据库存在 Samba 文件服务器上。

需求 2

为所有的用户创建帐号和目录，不分配 shell

市场部有 tom、jack，属于 sales 组

技术部有 red、blue，属于 tech 组

总经理 ceo、财务 fineance，属于 lead 组

需求 3

sales 组的文件只有 tom 和 leader 组的人可以读写，其它人只能看(非认证用户是不能进入目录的)；tech 组的文件只有 red 和 leader 组的人可以读写，其它人只能看(非认证用户是不能进入目录的)；

实施步骤

安装 samba34

服务器系统 FreeBSD 8.1-64bit

samba 版本 samba3.4

```
#cd /usr/ports/net/samba34
#make install clean
#cp
 /usr/local/share/examples/samba34/smb
 .conf.default /usr/local/etc/smb.conf
#echo 'nmbd_enable="YES"' >>
 /etc/rc.conf
#echo 'smbd_enable="YES"' >>
 /etc/rc.conf
#记得用>>追加符，不然你的 rc.conf 文件会被
 清零的:)
```

启动 samba

```
#/usr/local/etc/rc.d/samba start
```

先建立组和用户。把用户加入相应的组，给用户分配一个不可用的 shell；由于组比较多，我建议在纸上画好组织结构图，这样也方便自己理解。

添加用户组

```
# pw groupadd sales
# pw groupadd tech
# pw groupadd leader
```

然后用 pw groupshow 分别得知 sales、tech 及 leader 的 GID 分别为 1005、1006 和 1007

添加用户

```
# pw useradd tom -s /sbin/nologin -g
 1005
# pw useradd jack  -s /sbin/nologin -g
 1005
# pw useradd red -s /sbin/nologin -g
 1006
...
```

将其分别添加进 samba 用户库，此处的 samba 密码尽量选择不一样，并注意密码的复杂性。

```
# smbpasswd -a jack
New SMB password:
Retype new SMB password:
Added user jack.
...
```

建立 samba 的文件目录，并分配权限。

```
# mkdir /home/sales
# mkdir /home/tech
# ls -ld /home/sales
drwxr-xr-x  2 root  wheel  512 Jan 25
 05:36 /home/sales
# ls -ld /home/tech
drwxr-xr-x  2 root  wheel  512 Jan 25
 05:36 /home/tech
#给予 sales 和 tech 目录 777 权限的目的，是
 为了防止它们被本身的权限控制，用户能访问
 目录的最终权限是被文件目录的权限和 samba
 权限的最小权限交集所控制
# chmod 777 /home/sales
# chmod 777 /home/tech
```

给这个二个目录建议 sgid 权限，将能建立的文件都划归其组所有

```
# chgrp sales /home/sales
# chgrp tech /home/tech
# chmod g+s /home/sales
# chmod g+s /home/tech
[root@localhost samba]# ls -ld
 /home/sales /home/tech
# ls -ld /home/sales /home/tech
drwxrws---  2 root  sales  512 Jan 25
 05:36 /home/sales
drwxrws---  2 root  tech   512 Jan 25
 05:36 /home/tech
```

用 samba 的权限来严格限制这二个目录的权限

修改配置文件

```
vim /etc/samba/smb.conf
```

在文件的最后加入目录

```
[sales]
path = /home/sales
public = no
valid users = @sales,@leader
write list = tom,@leader
create mask = 0770
directory mask = 0770
[tech]
path = /home/tech
public = no
valid users = @tech,@leader
write list = red,@leader
create mask = 0770
directory mask = 0770
```

重启服务器

```
#/usr/local/etc/rc.d/samba start
```

总结

其实这里主要也只牵涉到三个知识点，我这里归纳下：

一、用户的最终权限分配为目录的权限和 samba 权限的最小交集；

二、valid 为认证用户，没经过认证的用户是不能看到其目录的文件的；

三、write list 会覆盖用户原有的用户权限，即使他们原先只有读的权限。▌※

本文为节选，完整内容请参考原文：
http://os.51cto.com/art/201101/244114.htm

公司目前所用远程备份主要是 vsftpd 和 rsync，故特将整个过程记录一下。

FreeBSD 8 下 Bridge 配置 openVPN

文/抚琴煮酒

公司环境介绍

公司的办公网络是 192.168.4.0/24，均只用单网卡 eth0，通过 Juniper 防火墙映射公网 IP 上网，即内网内所有的机器的网关均是防火墙的固定 IP-192.168.4.3,目前想通过外网拨号(VPN 服务器 IP 为 192.168.4.46)进公司内部局域网进行办公，另外还要求能够连通到公司内部的 192.168.10.0、192.168.20.0、192.168.21.0 网段的服务器，要求比之前复杂；而同事们的拨号环境又一样，有的是小区环境，有的是 ADSL，还有的是电信 3G 无线上网；我将大家的需求归纳了下，即：

一、要求能够在 ADSL 拨号或小区环境下顺利的 VPN 到公司的 openVPN 服务器中；

二、公司的办公环境是 192.168.4.0，拨上来要求能够连到 192.168.21.0,192.168.20.0,192.168.10.0 的服务器网段；

三、公司的网络环境已定型，不可能在路由器或防火墙上作更改；

四、考虑到路由穿透的问题，即最低限度要做到局对局。

基于以上考虑，我们准备采取 openVPN 的网桥模式。

公司的办公拓补很简单，如下图：

图 1 公司办公拓扑

以下资料参考 chinaunix 网友温占考的翻译文章，特此注明。

使用路由还是桥接的 VPN？

总的来说，路由对大多数人来说是一种更好的方式，因为它比桥接效率更高也更容易设置(仅 OpenVPN 配置文件本身)。路由还可以给每个客户端设置不同的访问权限。推荐使用路由，除非你需要使用依赖于桥接的特性，比如：VPN 需要处理非 IP 协议，例如 IPX 协议。在 VPN 上运行的程序依赖于网络广播(例如局域网游戏)。不建立 Samba 或 WINS 服务器，而允许在 VPN 上浏览 Windows 共享文件。

TUN 设备和 TAP 设备的不同？

TAP 设备是一块虚拟的以太网卡，TUN 设备是一个虚拟的点到点 IP 链接。

什么是桥接?

桥接是在一个子网上创建一个虚拟的、广域的以太网 LAN 一种技术。桥接的实践信息，参看 Ethernet Bridging Mini-HOWTO，形象的解释就是它就是连通不同局域网的桥梁当外网用户 VPN 过来时，如果能够连通 openVPN 机器，那么 openVPN 能够连通的机器，外网用户都应该能够连通。

桥接和路由的不同?

桥接和路由是通过 VPN 连接系统的两种方法。

桥接的优点：广播可以穿越 VPN--允许依赖局域网广播的软件运行，比如 Windows 的 NetBIOS 文件共享和网上邻居。无需配置路由：可以和以太网上的任何协议一起工作，包括 IPv4，IPv6，Netware IPX，AppleTalk 等等。

桥接的缺点：比路由效率低，扩展性不太好。

路由的优点：高效和可扩展、更好的 MTU 调节

路由的缺点：要使跨越 VPN 的网络浏览工作，客户端必须使用一个 WINS 服务器(比如 samba)。必须设置连接每一个子网的路由。依赖于广播的软件不能看到在 VPN 另一边的机器。仅支持 IPv4，如果连接两边的 tun 驱动明确支持 IPv6，则也支持 IPv6.

桥接和路由在配置上有什么不同?

当客户端通过桥接方式连接远端网络时，它被分配一个远端物理以太子网的 IP 地址，从而能够和远端子网其它机器交互就象它是连接在本地一样。桥接需要特殊的 OS-相关的工具用来将物理以太网卡和虚拟的 TAP 设备桥接起来。在 Linux 下，brtcl 就是这样一个工具。

桥接和路由很相似，主要的不同是路由的 VPN 不传送 IP 广播包，但是桥接的 VPN 传送 IP 广播包。

要使用桥接方式，连接的两端都必须使用--dev tap，如果使用路由方式，可以使用--dev tap 也可以使用--dev tun，但是连接的两端必须一致． 对于路由方式而言，--dev tun 的效率要更高一些。

桥接概览

以太网桥接将一个以太网接口和一个或多个虚拟 TAP 接口结合(combine)并将它们桥接为一个桥接接口。以太网桥接代表一个物理以太网交换机 (switch)的软件模拟，以太网桥可以 认为是在一台机器上共享一个 IP 子网连接多个以太网卡(物理的或虚拟的)的软件交换机，

通过将在不同地方的一个物理以太网卡和一个 OpenVPN 的 TAP 接口桥接，可以将两个以太网络在逻辑上合并为好象是一个以太子网

▌※

本文为节选，技术细节部分可参考原文：
http://network.51cto.com/art/201012/239786.htm

运维是实时对外提供服务的，置身于不安全的 Internet 中，相对于内网，安全隐患更多反而无故障率和故障修复时间更短，甚至不允许故障。

从网管转型到运维必须掌握的技能

文/Andy Feng

网管，也就是局域网管理员，外企称作 Helpdesk。运维工程师，主要负责外网生产服务器和内部开发、测试服务器的维护，简称 OP。

网管（Helpdesk）是很多希望从事 IT 行业的学生和工作者的第一份与 IT 有关的工作。网管这个职位含金量深浅不一，刚毕业的大学生可以做，有数十年经验的大牛也在做，主要区别在于

- 规模不同
- 自动化程度不同
- 安全需求程度不同

比如 10 台系统为 windows xp 的机器组成的工作组也是一个局域网；包含数千台电脑，各种操作系统，各种硬件架构，各种网络设备各种网络协议也叫局域网；但是管理这两个局域网的 helpdesk 能力却有天壤之别。

OP 这个词语代表的意思很多，这个简称来自于英文的 Operations 一词。OP 工作内容主要就是维护公司的服务器能够正常提供服务细分的话包括系统，网络，应用，数据库，具体根据公司的规模和职位职能不同，运维的定义也不同。现在市面上主要的 OP 有三种：网络游戏运维，网站运维，大型项目测试和生产环境运维。

网管和运维都涉及到网络，系统，数据库，应用软件，硬件等等，但内网的网管更偏向网络和硬件，外网运维更偏向架构和安全。

运维是实时对外提供服务的，置身于不安全的 Internet 中，相对于内网，安全隐患更多，反而无故障率和故障修复时间更短，甚至不允许故障。

所以，从网管转型为运维需要什么？

1、一种或者多种非常熟悉的服务器操作系统，推荐 Linux。

2、熟悉常见服务器的搭建及优化，比如 HTTP，DHCP，CACHE，MONITER，FTP，SSH 等。

3、熟悉一种脚本语言，提高自己自动化处理能力，减轻工作压力，推荐 bash shell。

4、安全知识，了解基本的系统安全、网络安全、WEB 安全的攻击与防御。

上面谈到都是技术需求，还有就是素质需求。要有随时加班，随时处理故障的觉悟，要有细心负责的工作态度，还要有一颗强健的心脏，在出现重大事故的时候，能够妥善处理，把事故的损害降到最低。▌※

原文：
http://os.51cto.com/art/201009/224764.htm

如果你是个较真的系统管理员且确实需要监控用户的活动，那就写个内核模块记录所有用户的键盘记录，并根据 uid 或其他参数进行过滤。

用户行为监控：bash history logging 攻防

文/ithilgore
编译/高永伟

Bash 堪称*nix 世界使用最广泛的 shell，其特性之一是历史命令(history)机制。此机制主要用于为用户提供方便 - - 少敲几下键盘提高工作效率。然而，被广泛讨论的是 bash_history 可以用作 logging 机制以此来监控用户的活动。此文将对上述问题进行讨论并解释为啥 logging 机制在少数人面前会失效。我们将见到各种用于保护 history 文件的防御措施是如何不费吹灰之力或稍微费点力就被突破的。随着讨论的跟进，突破的限制也将变得更严，但这并不代表突破起来就更困难，与之相反大部分方法都是可以不费脑子的最后，我们将修改 bash 的源码来实现”无敌”logging 机制，也将看到”无敌”并不是真正的无敌。

加固 bash_history

假设你所管理的系统提供 shell 登录功能，你的用户当中有个别及其讨人厌的家伙，于是你想监控他的活动，因为你非常怀疑他半夜三更使用你所负责保护的 CPU 和系统资源作恶意行为（或是其他的，例如下毛片等）。我们暂且叫他二哥（此处原文为 Bob，Bob 一名在国外经常用来指代坏蛋）。

因为所有用户都是使用 bash 作为默认 shell，你开始着手修改 bash 的配置文件：

第 1 步：使 bash 历史记录文件和相关文件无法被删除或修改。

二哥所做的第一件事应该是建立 history 到/dev/null 的链接。

```
bob$ rm ~/.bash_history
bob$ ln -s /dev/null  ~/.bash_history
```

这可以通过修改历史记录文件为只能被追加来进行阻止，执行以下命令来改变其属性：

```
# chattr +a /home/bob/.bash_history
```

这是使用文件系统附加属性来指定文件只能被追加，大多数文件系统支持此功能(例如 ext2/3,XFS,JFS)。在 FreeBSD 上可以执行：

```
# sappnd /home/bob/.bash_history
```

你还应修改 shell 启动相关的其他文件的这个属性：

```
# chattr +a /home/bob/.bash_profile
# chattr +a /home/bob/.bash_login
# chattr +a /home/bob/.profile
# chattr +a /home/bob/.bash_logout
# chattr +a /home/bob/.bashrc
```

前三个文件在交互式 bash shell（或非交互式 sehll 使用–login 选项）调用时被读取（在读完全局配置文件 /etc/profile 后）。.bashrc 文件只在当 non-login 交互

式 shell 调用时被读取。这意味着当二哥已登进系统后，用以下方法自己调用一个新 shell 时：

```
bob$ bash
```

此时只有.bashrc 文件被读取，而上面所列的前三个配置文件不会再次被读取了。

做了以上属性的修改后再来做更进一步的”加固”，一个所谓的保护措施。

第 2 步：配置 .bash*配置文件

所有的配置将针对.bashrc 文件，因为其他三个配置文件本身会调用.bashrc，也就是说.bashrc 无论如何都会被读取（不管用户是否刚登录或是登录后手工调用 bash shell）。

所以，所有修改都针对.bashrc 的好处是可以防止二哥登录后手工调用新的 bash shell 来跳过仅在

```
.bash_profile,.bash_login,.profile
```

三个配置文件中生效的配置选项，另一好处是这三个文件本身都会调用.bashrc，所以在首次登录系统时.bashrc 当中的配置也会生效。

```
# cat >> /home/bob/.bashrc << EOF
> shopt -s histappend
> readonly PROMPT_COMMAND=”history -a”
> EOF
```

此处 histappend 选项的作用是让 bash 附加上最后一行$HISTSIZE 给$HISTFILE 文件（一般是~/.bash_history 文件），不管交互式 shell 何时退出。默认的，bash 每次均会覆盖$HISTFILE 以保证只有一个 session 被保存以此来节约空间。

环境变量 PROMPT_COMMAND 会保存一条将被优先执行的命令，意思是说” history -a”命令将在用户执行命令前被优先执行，这将保证不管当前命令前一条是执行的什么，它将立即被追加进$HISTFILE，而不用等待整个会话结束再将历史命令从内存记录至硬盘。

此处的 readonly 作用是使变量不可修改以防止被二哥覆盖掉或是直接屏蔽掉。

最后要完成的步骤是使所有与 bash_history 相关的环境变量都变为 readonly：

```
readonly HISTFILE
readonly HISTFILESIZE
readonly HISTSIZE
readonly HISTCMD
readonly HISTCONTROL
readonly HISTIGNORE
```

第 3 步：禁掉系统中所有其他 shell，一般包括 csh,tcsh,ksh。

```
# chmod 750 csh
# chmod 750 tcsh
# chmod 750 ksh
```

这将阻止二哥把 bash shell 切换成其他 shell。

现在，机敏点的管理员会抱怨上面的都是 shit！

这是怎么回事呢？由于篇幅所限，这里就不详细介绍了，感兴趣的读者们可以通过以下链接继续阅读。▌※

英文原文：
http://sock-raw.org/papers/bash_history

译文全文：
http://os.51cto.com/art/201102/244661.htm

推荐阅读：

Linux 下的实用 bash 命令分享

假如想要发现哪个进程使用了大量内存的话，我通常会使用 htop 或 top 而非 ps 。

系列连载：最牛 B 的 Linux Shell 命令（3）

文/Peteris Krumins
编译/BOY PT

1. 更友好的显示当前挂载的文件系统

```
mount | column -t
```

这条命令适用于任何文件系统，column 用于把输出结果进行列表格式化操作。

下面是单单使用 mount 命令的结果：

```
$ mount
/dev/root on / type ext3 (rw)
/proc on /proc type proc (rw)
/dev/mapper/lvmraid-home on /home type ext3 (rw,noatime)
```

而加了 column -t 命令后就成为这样了：

```
$ mount | column -t
/dev/root                on  /      type  ext3   (rw)
/proc                    on  /proc  type  proc   (rw)
/dev/mapper/lvmraid-home  on  /home  type  ext3   (rw,noatime)
```

另外你可加上列名称来改善输出结果

```
$ (echo "DEVICE - PATH - TYPE FLAGS" &&
 mount) | column -t
DEVICE                   -   PATH   -     TYPE   FLAGS
/dev/root                on  /      type  ext3   (rw)
/proc                    on  /proc  type  proc   (rw)
/dev/mapper/lvmraid-home  on  /home  type  ext3   (rw,noatime)
```

列 2 和列 4 并不是很友好，我们可以用 awk 来再处理一下

```
$ (echo "DEVICE PATH TYPE FLAGS" &&
 mount | awk '$2=$4="";1') | column -t
DEVICE                   PATH   TYPE   FLAGS
/dev/root                /      ext3   (rw)
/proc                    /proc  proc   (rw)
/dev/mapper/lvmraid-home  /home  ext3   (rw,noatime)
```

最后我们可以设置一个别名，为 nicemount

```
$ nicemount() { (echo "DEVICE PATH TYPE
 FLAGS" && mount | awk '$2=$4="";1') |
 column -t; }
```

自己试试 nicemount 吧！

2. 运行前一个 Shell 命令，同时用“bar”替换掉命令行中的每一个“foo”

```
!!:gs/foo/bar
```

!! 表示重复执行上一条命令，并用 :gs/foo/bar 进行替换操作。

3. 实时某个目录下查看最新改动过的文件

```
watch -d -n 1 'df; ls -FlAt /path'
```

watch 是实时监控工具，-d 参数会高亮显示变化的区域，-n 1 参数表示刷新间隔为 1 秒。

df; ls -FlAt /path 运行了两条命令，df 是输出磁盘使用情况，ls -FlAt 则列出 /path 下面的所有文件。

ls -FlAt 的参数详解：

```
-F 在文件后面加一个文件符号表示文件类型，
 共有 /=>@| 这几种类型， 表示可执行文件，/
 表示目录，= 表示接口( sockets) ，> 表示
 门， @ 表示符号链接， | 表示管道。
-1 以列表方式显示
-A 显示 . 和 ..
-t 根据时间排序文件
```

4. 通过 SSH 挂载远程主机上的文件夹

```
sshfs name@server:/path/to/folder
 /path/to/mount/point
```

这条命令可以让你通过 SSH 加载远程主机上的文件系统为本地磁盘，前提是你需要安装 FUSE 及 sshfs 这两个软件。

卸载的话使用 fusermount 或 umount 命令：

```
$ fusermount -u /path/to/mount/point
# umount /path/to/mount/point
```

5. 通过 DNS 来读取 Wikipedia 的词条

```
$ dig +short txt <keyword>.wp.dg.cx
```

这也许是最有趣的一条技巧了，David Leadbeater 创建了一个 DNS 服务器，通过它当你查询一个 TXT 记录类型时，会返回一条来自于 Wikipedia 的简短的词条文字，这是他的介绍。

这里有一个样例，来查询 “hacker” 的含义：

```
$ dig +short txt hacker.wp.dg.cx
"Hacker may refer to: Hacker (computer
 security), someone involved
in computer security/insecurity, Hacker
 (programmer subculture), a
programmer subculture originating in
 the US academia in the 1960s,
which is nowadays mainly notable for
 the free software/” “open
source movement, Hacker (hobbyist), an
 enthusiastic home computer
hobbyist http://a.vu/w:Hacker"
```

这里使用了 dig 命令，这是标准的用来查询 DNS 的系统管理工具，+short 参数是让其仅仅返回文字响应，txt 则是指定查询 TXT 记录类型。

更简单的做法是你可以为这个技巧创建一个函数：

```
wiki() { dig +short txt $1.wp.dg.cx; }
#然后试试吧:
wiki hacker
"Hacker may refer to: Hacker (computer
 security), …"
```

如果你不想用 dig ，也可以用 host 命令：

```
host -t txt hacker.wp.dg.cx
```

另外在 Twitter 上看过某人的创意，用普通的 dns 来作为程序版本更新的查询服务器：设定域名

software-version-check.example.com

的 A 记录为 1.2.40.3，对比自己的版本号，嗯，有更新了！

6. 用 Wget 的递归方式下载整个网站

```
wget --random-wait -r -p -e robots=off
 -U Mozilla www.example.com
```

参数解释：

```
- random-wait 等待 0.5 到 1.5 秒的时间来
 进行下一次请求
-r 开启递归检索
-e robots=off 忽略 robots.txt
-U Mozilla 设置 User-Agent 头为 Mozilla
```

其它一些有用的参数：

```
- limit-rate=20K 限制下载速度为 20K
-o logfile.txt 记录下载日志
-l 0 删除深度（默认为 5）
-wait=1h 每下载一个文件后等待 1 小时
```

7. 复制最后使用的命令中的参数

```
<Ctrl + .> or <ESC + . >
```

这个快捷键只能工作于 shell 的 emacs 编辑模式，它可以从最后使用的命令行中复制参数到当前命令行中，下面是一个样例：

```
$ echo a b c
a b c
$ echo <Press ALT + .>
$ echo c
```

你可以重复执行该快捷键，以便获取自已需要的参数，

以下是样例：

```
$ echo 1 2 3
1 2 3
$ echo a b c
a b c
$ echo <Press ALT + .>
$ echo c
$ echo <Press ALT + .> again
$ echo 3
```

另外，假如你想指定第 1 个或第 2 个，或者是第 n 个参数的话，可以按 ALT + 1（或 ESC + 1）或 ALT + 2（或 ESC +2）这样形式的快捷键。

以下是样例：

```
$ echo a b c
a b c
$ echo <Press ALT + 1> <Press ALT + .>
$ echo a
a
$ echo <Press ALT + 2> <Press ALT + .>
$ echo b
b
```

8. 执行一条命令但不保存到 history 中

```
$ <space>command
```

这条命令可运行于最新的 Bash shell 里，在其它 shell 中没测试过。

通过在命令行前面添加一个空格，就可以阻止这条命令被保存到 bash history (~/.bash_history) 文件中，这个行为可以通过 $HISTIGNORE shell 变量来控制。我的设置是 HISTIGNORE="&:[ ]*"，表示不保存重复的命令到 history 中，并且不保存以空格开头的命令行。$HISTIGNORE 中的值以冒号分隔。

如果你的命令内包含密码，比如 mysqladmin，不把它记录在历史当中是好主义。

9. 显示当前目录中所有子目录的大小

```
du -h --max-depth=1
```

–max-depth=1 参数可以让 du 命令显示当前目录下 1 级子目录的统计信息，当然你也可以把 1 改为 2，进一步显示 2 级子目录的统计信息，可以灵活运用。而 -h 参数则是以 Mb 、G 这样的单位来显示大小。

译者注：在此推荐一个小工具 ncdu，可以更方便的达到此效果。

10. 显示消耗内存最多的 10 个运行中的进程，以内存使用量排序

```
ps aux | sort -nk +4 | tail
```

显然这并不是最好的方法，但它用着还不错。

这是一个典型的管道应用，通过 ps aux 来输出到 sort 命令，并用 sort 排序列出 4 栏，再进一步转到 tail 命令，最终输出 10 行显示使用内存最多的进程情况。

假如想要发现哪个进程使用了大量内存的话，我通常会使用 htop 或 top 而非 ps 。▌※

本文节选自英文原文：
http://www.catonmat.net/blog/another-ten-one-liners-from-commandlinefu-explained

译文全文：
http://www.isspy.com/most_useful_linux_commands_3/

招募启事

《Linux 运维趋势》的建设需要您的加入！

您可以通过如下方式参与我们杂志的建设：

1、推荐文章

无论是您在互联网上看到的好文章，还是您自己总结/整理的资料；无论是英文还是中文；无论是入门的还是高端的，都欢迎推荐！推荐方式包括：

a）在技术圈中分享：http://g.51cto.com/linuxops

b）发邮件给编辑：yangsai@51cto.com

2、投稿

如果您认为自己在 Linux 方面具有专家级别的能力，并且有与大家分享您技术经验的热诚，同时也有兴趣挣点稿费花花，那么欢迎您的投稿！

如果您在 IT 技术方面的翻译有很高的能力，能够快速、高质量的完成译文，并且也经常浏览到一些 Linux 方面的优秀外文，那么也欢迎您的投稿，或加入我们 51CTO 的翻译团队！

投稿信箱：yangsai@51cto.com

3、推广与意见

如果您喜欢我们的杂志，认为这本杂志对于您的工作有所帮助，请向您的 Linux 好友、同事们推荐它！

如果您觉得这份杂志还有什么地方需要改进或补充，也希望您能够提出您的宝贵意见！

联系人：yangsai@51cto.com

下期预告

下期主题为：迁移备份。欢迎关注！

本刊为月刊，预定每月发布日期为：

每个月的第二个星期五

您可以通过如下方式检查新刊发布：

1、通过电子邮件订阅：

订阅方式请参考本杂志的专题页：
http://os.51cto.com/art/201011/233915.htm

2、经常光顾 51CTO Linux 频道：
http://os.51cto.com/linux/