deepin 集结

05 期
2017 年 03 月

深度操作系统公众号

卷首语

深度科技副总经理
王棣

2002年上映了一部由香港最著名的搞笑女星吴君如主演的电影《金鸡》，影片通过吴君如饰演的妓女“阿金”的口中，道出了香港从1970年代末至2000年代初的社会变迁。不论是像阿金这样生活在社会最底层的妓女身份的小老百姓，还是片中的“陈教授”，抑或当了“接盘侠”的“Richard”，都在大时代的起起伏伏中艰难求生。生存不易，但是他们都对生活充满了希望，即使在最困难的时候，也从来不放弃继续的勇气。

2017年恰逢是鸡年，对于深度人来说将会是迎来一个大的变革的年份。我们不妨从阿金的身上学习到这样一份精神，每个人在新的一年里踏踏实实的用心去做好自己份内一点一滴的事，积少成多，必将会让深度和我们自己在这个机遇和挑战并存的时代里，生机勃发。

借用《金鸡》中Richard唱给阿金的几句歌词献给新年里所有深度的伙伴们：

希望之光，在空中闪耀；
一颗小星，照亮天上路。
跨过整个大地，开展新的黎明。

deepin集结

策划 **Hosted by**
武汉深之度科技有限公司 Wuhan Deepin Technology Co., Ltd.
编辑 **Edited by**
《deepin集结》杂志编辑部 Editorial Office of DEEPIN JIJIE

总编辑 **Editor-in-chief**
刘闻欢 Liu Wenhuan
副总编 **Deputy Editor**
许　可 Xu Ke
执行编辑 **Executive Editor**
郝　俊 Hao Jun
编　辑 **Editor**
许　峰　张文豪 Xu Feng　Zhang Wenhao
脱文剑 Tuo Wenjian
采　编 **Assistant Editor**
张凤玲 Zhang Fengling
李会会　蒋　文 Li Huihui　Jiang Wen
美术设计 **Art Editor**
秦　娣 Qin Di

网　站 **Website**
http://www.deepin.com
邮箱投稿 **Contribution**
deepin-magazine@deepin.com
市场推广 **Marketing**
account-marketing@deepin.com

武汉联络处 **Wuhan Office**
地址 **Address**
武汉市光谷大道77号 The 6th Floor, Building B18, FinancialHarbour,
光谷金融港B18栋6楼 No. 77, Optics Valley Avenue, Wuhan, China
邮　编 430223
电　话 +86-27-87805607

北京联络处 **Beijing Office**
地址 **Address**
北京市西城区新街口外大 Room 603, Tower B, Xinwai Street No.28,
街28号普天德胜B座603室 XiCheng District, Beijing, China
邮　编 100088
电　话 +86-10-62669499

上海联络处 **Shanghai Office**
地址 **Address**
上海市长宁区 Room 15A01, No. 1258, Yuyuan Road,
愚园路1258号15A01室 Changning District, Shanghai, China
邮　编 200050
电　话 +86-21-60726030

准印证号 （鄂）4300107
承印单位 武汉金港彩印有限公司
出版日期 2017年03月

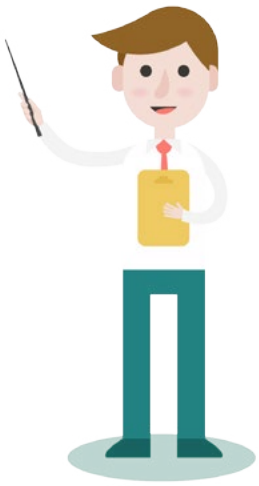
目录

季刊・总第 5 期
2017 年 03 月

深度操作系统 15.4 给你“好看”

深度操作系统 15.4 全新设计了控制中心，并默认预置了深度家族的一系列应用，不仅做到好用还要好看，还提升了用户体验。

深度科技2016大事记



2016 年，深度科技走过了艰难而充实的一年，接下来，我们一起细数过去一年深度科技大事记，共同展望 2017。

2016 01 月

- 深度科技官方网站正式上线，方便用户更加全面的了解深度科技。
- Deepin 家族又添新成员，manjaro 宣布发布基于 Deepin 的 manjaro-deepin 发行版。

2016 02 月

- 深度科技内刊杂志《deepin 集结》正式创刊，成为代表公司形象建立的窗口之一。

2016 03 月

- 中国网络空间安全协会成立，深度科技为发起单位之一。
- 深度科技联合恒银金融加速推动银行业国产化进程。
- 深度科技为唯一上榜操作系统企业——2016 最具影响力国产开源软件 TOP100 榜单。

2016 04 月

- 深度科技作为在信息安全中承担重要角色的操作系统研发企业参与“4.29 首都网络安全日”。

2016 05 月

- 通过 CMMI3（能力成熟度模型集成）等级认证。
- 深度操作系统产品发布会成功举办，正式发布深度操作系统 V15 版本。
- 深度科技与金山软件签订战略合作协议，双方就打造国产应用软件生态方面展开全方位合作与交流。

- 深度科技联合网易云音乐合作发布网易云音乐 Linux 版，推出国内首款 Linux 版在线音乐应用。
- 深度科技亮相第二十届中国国际软件博览会，展示国产操作系统最新研发成果。
- 服务器操作系统版本 V15.0 版本亮剑，正式投入市场。

2016 06 月

- 深度科技加入中国开源云联盟，携手共同推进开源软件和云计算事业发展。
- 深度科技受邀参加第十一届开源中国开源世界高峰论坛，并与 Intel 软件与服务事业副总裁 Mauri Whalen 就合作事宜进行会谈。

2016 07 月

- 深度应用家族又添新成员——深度云扫描，解决用户在 Linux 环境下的扫描难题。

2016 08 月

- Linux 版招行网银专业版正式上线，解决困扰 Linux 用户多年无法在 Linux 系统下正常使用网银的痛点。
- 深度看图 V1.0 发布，外观时尚、性能流畅，支持多种图片格式。

2016 9-10 月

- 2016 第六届深度开发者与用户大会（DDUC）在西安、北京、武汉、上海、广州 5 座城市成功举办 6 场线下开发者与用户交流大会。
- 深度科技作为武汉市重点企业亮相网络安全博览会，向外界展示深度科技作为操作系统研发企业，竭力为国产化及网络信息安全做出贡献。
- 深度终端 V2.0 发布，速度快、高颜值、资源占用少、极致的终端体验。

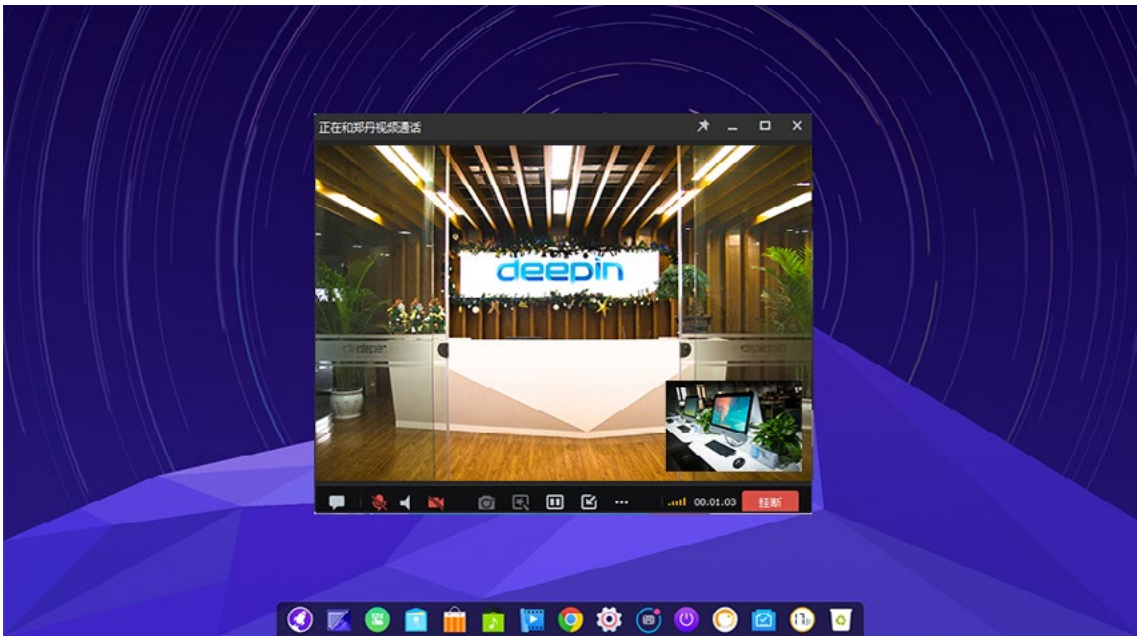
2016 11 月

- 深度科技荣获“楚天杯”工业设计大赛的“软件与交互产品设计”项，并获得该项设计的一等奖。
- 服务器操作系统版本 V15.1 版本发布，集成最新安全补丁，在批量部署支持，安全等方面均有所加强。

2016 12 月

- 深度科技作为优秀版权示范企业参加“第六届中国国际版权博览会”。
- Linux 操作系统首次支持 QQ 视频通话，用户可完美体验 Linux 平台下新版 QQ 的全部功能。

这一年还发布了深度操作系统 V15 系列版本，不断地突破自我，带给用户更为友好、美观、易用的操作系统；参与行业内展会，与行业内软硬件厂商做更多交流；与合作伙伴展开更为全面的合作，推动国产操作系统生态环境的发展 ..... 深度科技 2016 年的答卷如上，我们期待着更好的 2017，这一路时光漫长，携手相伴，感恩有你 ，Hello 2017。



喜大普奔：Linux 操作系统首次支持 QQ 软件全功能使用

深度科技 市场部 / 文

腾讯 QQ 作为一款即时通信的热门软件，是当下人们工作与生活中必不可少的聊天工具，该软件支持在线聊天、视频通话等功能。在 Windows 平台下可畅通运行，然而 Linux 操作系统平台用户却一直无法完美体验这款基本的聊天软件，在以往使用时，会产生诸多不稳定现象，点击 QQ 视频时更会导致软件崩溃。不能使用 QQ 软件进行视频聊天一直是 Linux 操作系统用户的一大难关。

攻克 Linux 平台下 QQ 软件的最后一道难关

近日，国产操作系统研发企业——深度科技团

队，为了让用户完美体验 Linux 平台下全功能的新版 QQ，对 QQ 重新进行打包，修复以往用户反馈的问题，也攻克了 QQ 在 Linux 平台下最后一道难关——视频通话。

采用 wine 技术实现 API 支持 Wndows 接口，从而支持 Windows 应用能够在 Linux 下运行。其中摄像头采用 DirectShow 技术；视频截取采用 Linux 的 API 是 V4L(Video4Linux)，也解决了部分调用 V4L 的 API 导致的问题。

通过与深度科技研发团队了解，在研发过程中遇到了多重问题，最终克服困难一一解决：

1、打开摄像头设备占用的问题

通过由简入繁的方法，先通过解决一些小的 demo 程序的问题，然后再来解决 QQ 的问题。对比 QQ 和 demo 程序，发现 QQ 和其它程序不同，有多次打开摄像头的操作，但是由于对象释放的问题导致后面打开摄像头出现设备占用打开失败。

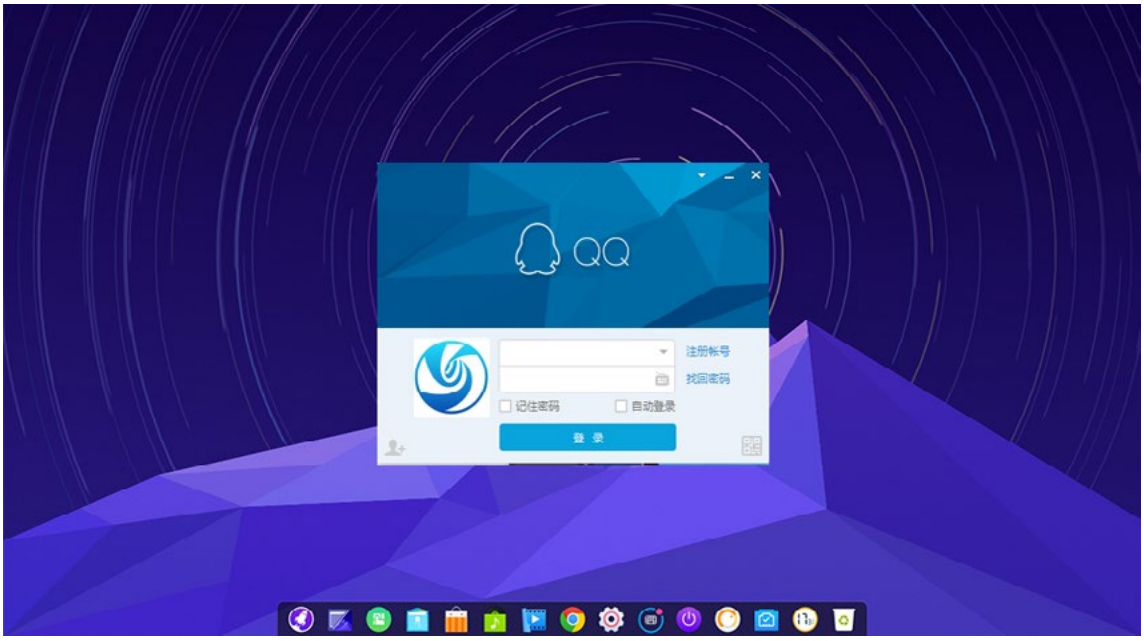
2、获取不到摄像头数据的问题

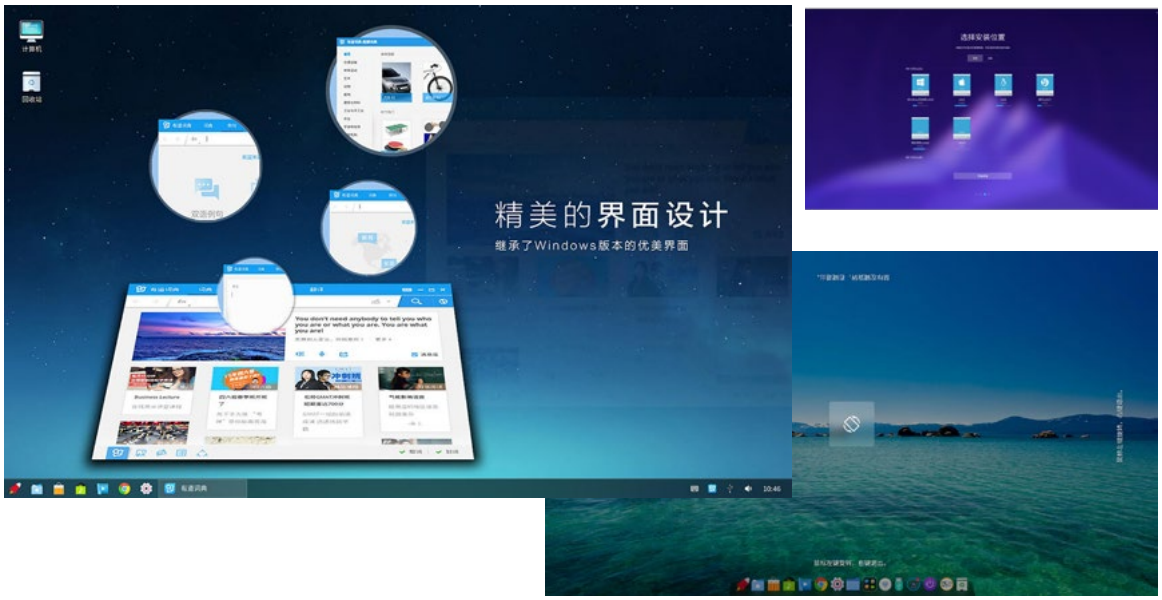
在解决部分应用和 demo 程序摄像头问题之后，发现 QQ 视频还是不能正常使用，读不到数据并且读取数据线程和控制线程还会死锁。通过 DirectShow 的问题，分析 DirectShow 的状态机制，解决 QQ 摄像头控制和读取数据的问题。

3、关闭摄像头出现卡死的问题

完善线程同步代码，解决读取数据线程和控制线程死锁的问题。

Linux 用户无需安装双系统，无需使用虚拟机，且在没有性能损耗的前提下，即可在 Linux 操作系统上完美运行 QQ 软件，进行视频通话。深度操作系统是首个支持 QQ 软件全功能使用的国产





Linux 发行版，这在 Linux 平台上是一个全新的突破，不仅解决了一直以来困扰用户不能视频通话的问题，也使国产操作系统在全生态建设与兼容 Windows 系统软件上有了进一步发展。

掌握先进技术 推动国产化应用

深度科技作为国产操作系统研发团队，以提供安全可靠、美观易用的国产操作系统与开源解决方案为目标，公司拥有顶尖的操作系统研发、行业定制、支持服务与培训等多方面专业人才，能够满足不同用户和应用场景对操作系统产品的广泛需求。

根据市场研究公司 Net Applications 最新数据显示，2016 年 6 月份 Windows 操作系统的总份额为 89.79%，Linux 的市场份额为 2.02%。显而易见，在全球市场方面，Linux 操作系统依旧不容乐观，只有更多的全球软硬件厂商投入 Linux 生态建设，才能扩大开源软件在普通用户中的使用率。

近年来，深度操作系统发展迅速，获得全球四十多个国家用户的支持，累计下载量数千万次，并成为在 Distrowatch 上排名最高的中国 Linux 操作系统发行版。无论是新版 QQ，还是早在之前发布的招商网银等应用，都推动了 Windows 应用向 Linux 操作系统的迁移，同时 Linux 版本的网易云音乐、搜狗输入法、有道词典、WPS 等原生应用，也解决了越来越多 Linux 用户的需求。在今年，深度操作系统凭借自身在 Linux 领域的快速发展，入选互联网周刊登出的“2016 最具影响力国产开源软件 TOP100”，是唯一上榜的操作系统企业。

深度科技始终在朝立足中国、面向国际、能够代表中国技术水平一流的操作系统企业方向发展，将掌握的开源技术投入到市场化运作中，发挥国产基础软件领域技术、市场、服务的引导作用，推动国产操作系统的广泛应用与创新。

深度科技总经理刘闻欢荣获
“2016 首届中国光谷最具新锐力企业领导者”奖项

深度科技 市场部 / 文

2017 首届中国（武汉）新经济发展高峰论坛暨 2016 首届中国光谷新经济年度人物颁奖活动在湖北武汉东湖国际会议中心举行。活动现场揭晓了荣获“2016 首届中国光谷新经济年度人物”、“2016 首届中国光谷最具新锐力企业领导者”、“2016 首届中国光谷年度致敬企业家”，并进行现场颁奖及致辞。

深度科技总经理刘闻欢凭借多年来出色的企业领导力，荣获“2016 首届中国光谷最具新锐力企业领导者”奖。他带领深度科技由社区研发团队历经波折困难逐渐成长为中国最具有代表性国产操作系统研发的中坚企业，将深度科技操作系统产品向党政军、金融、运营商、教育等多行业推广、应用，在国产化进程中迈向更高的一步。

组委会给“2016 首届中国光谷最具新锐力企业领导者”的颁奖词为：

他们站在被看好的未来技术和投资风口
代表着再造硅谷的颠覆性力量
代表着光谷的兴奋和期待

在本次论坛上，除了部分获奖企业家的主题演讲外，还有来自《硅谷百年史》作者皮埃罗·斯加鲁菲（Piero Scaruffi）在会上做了“百年创新，如何在中国打造第二个硅谷”的主题演讲。

深度操作系统入选“2016 年度最受欢迎中国开源软件 TOP 20”

深度科技 市场部 / 文

每年年底，开源中国网站都会评选最受欢迎软件榜单，该榜单根据软件在社区里的用户关注度、活跃度、访问量等信息来对受欢迎程度进行量化，然后投票选出最终结果。2016 年度最受欢迎中国开源软件评选结果已出炉，深度操作系统作为国产的开源操作系统入选，名列 18 位，是国内唯一上榜的操作系统产品。

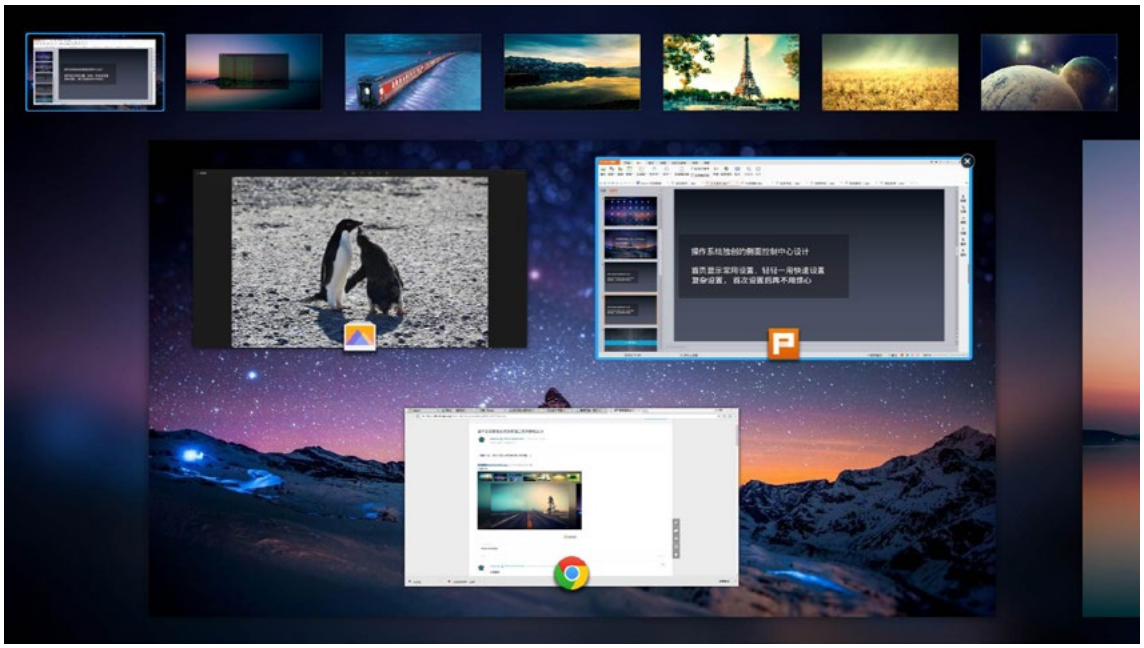
深度操作系统项目：

深度操作系统是国内最流行和活跃的 Linux 发行版本，是由深度科技自主开发的美观易用、安全可靠的操作系统。其桌面环境主要由桌面、启动器、任务栏、控制中心、窗口管理器等组成，系统中预装了 WPS Office、搜狗输入法、有道词典、网易云音乐以及深度特色应用。

深度科技团队非常注重友好的体验和美观的设计，

2016 年度最受欢迎中国开源软件评选结果：

TOP20 (参与用户数：17114)		刷新
1.	JFinal	4092 票
2.	ECharts	3956 票
3.	Layui	3353 票
4.	Druid	2707 票
5.	Vue.js	2548 票
6.	fastjson	2389 票
7.	Dubbo	2137 票
8.	ThinkPHP	1764 票
9.	JEECG	1467 票
10.	zTree	1454 票
11.	Mycat	1426 票
12.	禅道	1144 票
13.	Apache RocketMQ	1083 票
14.	JFinal Weixin	975 票
15.	JeeSite	853 票
16.	WeUI	840 票
17.	sharding-jdbc	813 票
18.	Deepin	808 票
19.	JeeWx 捷微	797 票
20.	Nutz	583 票



因此对于大多数用户来说，既能在深度操作系统上体验到丰富多彩的娱乐生活，也可以很好的代替 Windows 系统满足日常工作需求。

国产操作系统新高度——深度操作系统
支持全球 30 多种语言
累计下载量已达 5000 万次
位列全球关注度排行榜前 20 名

近年来，深度科技与搜狗、WPS、网易等合作伙伴进行了多方位合作，共同打造国产操作系统生态。同时，还努力解决迁移 Windows 平台软件带来的各种兼容性问题，以便用户平滑的过渡到开放安全的 Linux 平台上来。

深度科技握手龙芯中科

◉ 深度科技 市场部 / 文

近日，深度科技与龙芯中科在北京签署了战略合作协议，这为 2017 年的新春增添了一抹靓丽的色彩。

从 2015 年起，深度科技就已经开始将操作系统移植到龙芯平台上，并逐步完成了相关的适配优化工作，为数个项目打造了良好的基础。

2017 年 2 月，双方为下一阶段的合作在龙芯中科总部进行了探讨，就目前双方的产品模式和企业理念都做出了详尽的解读。在这次会议中，深度科技总经理刘文欢介绍了深度操作系统在内核优化、驱动支持、应用开发、交互设计等方面的工作。龙芯中科总裁胡伟武也就其未来的发展路线和最新型号的龙芯 CPU 进行了介绍。

经过这次交流，双方一致认为：应继续坚持走自主发展的道路，在充分与上下游合作伙伴进行适配的同时，建立完整的基于“龙芯 + 深度”的生态链。

2017 年是国产自主可控发展的重要之年，在此

深度桌面操作系统龙芯版

契机下，双方通过战略合作协议的签署，再次明确了未来的合作模式，合作方向及行动的路线和目标。相信深度科技和龙芯中科会在这样的东风下，为中国在基础软件和核心芯片领域的自主可控发展添砖加瓦。

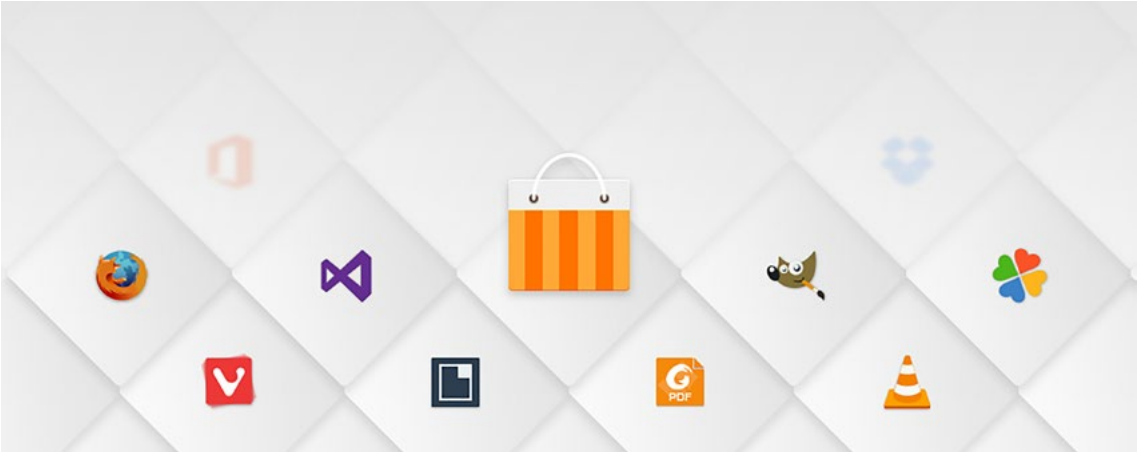
关于深度科技

深度科技是国内顶尖的 Linux 研发团队，基于 Linux 内核的深度操作系统是公司的主要产品。公司拥有系统研发、软件包维护、内核开发、国际化、交互设计等专业人才，能够满足不同用户对操作系统定制化和研发的需求。

关于龙芯中科

龙芯中科面向国家信息化建设的需求，面向国际信息技术前沿，以安全可控为主题，以产业发展为主线，以体系建设为目标，坚持自主创新，掌握计算机软硬件的核心技术，为国家安全战略需求提供自主、安全、可靠的处理器，为信息产业及工业信息化的创新发展提供高性能、低成本、低功耗的处理器。

网友留言



龙芯深度应用商店 V1.0 发布

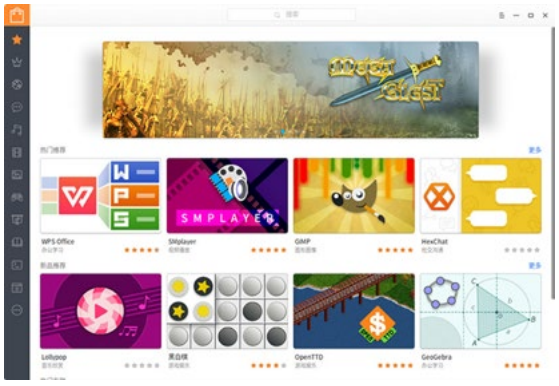
深度科技 / 文

经过深度科技和龙芯中科的合作努力，龙芯版深度应用商店 v1.0 发布啦！龙芯版深度应用商店旨在更好地完善“龙深”生态链，目前已经集成了超过 300 款的各类应用软件，以此满足日常使用者的常规需求。应用商店的发布也希望更多软件开发商加入，为大家提供更多精品应用。

龙芯版深度应用商店是一款集应用展示、下载、安装、评论、评分于一体的应用程序。深度商店为您精心筛选和收录了不同类别的应用，同时每款应用都经过大量安装测试验证，您可以进入商店搜索热门应用，一键下载并自动安装。

界面简洁，交互友好

深度商店以极简、扁平化的设计风格，呈现给大家简约精致的外观，应用详情页下载、更新及打开三种状态一目了然，给您完美的体验。



分类导航，精准推荐

截止到 2017 年 3 月 1 日，龙芯版深度应用商店至今已经收录了 335 款应用，按照各个应用的功

能分为十一个类别导航。其中：

网络应用类 20 款

社交沟通类 6 款

音乐欣赏类 18 款

视频播放类 23 款

图形图像类 23 款

游戏娱乐类 86 款

办公学习类 36 款

阅读翻译类 12 款

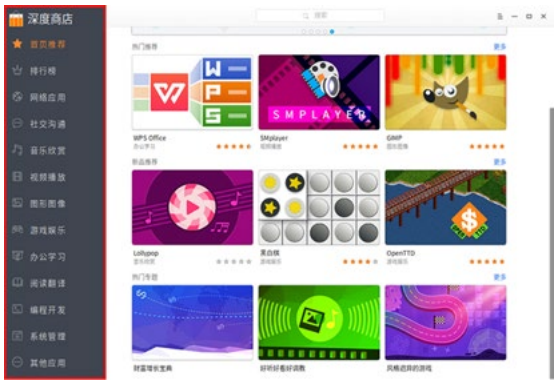
编程开发类 36 款

系统管理类 61 款

其他应用类 14 款

分类的显示效果清新、直观，协助你快速的定位到想要的应用。

商店主页分为轮播图、热门推荐、新品推荐和热门专题，为您呈现最新、最热的精品应用，以专业的态度给你列明尝鲜菜单。应用专题将以独立页面的形式展现，以获得更好的显示效果！



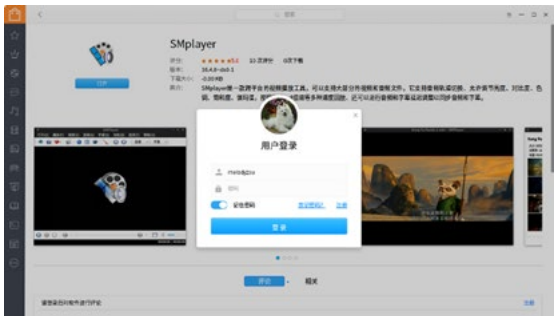
精挑细选，品质保证

深度商店中每一款应用都由专人负责检查和评估，为大家挑选上架精品应用。另外，商店的应用封面、应用图标、应用描述等信息，均由设计师与文档工程师精心制作，搭配丰富、精美的应用截图，让您更快“淘”到心仪的应用。

优化体验，细致入微

专注提供精品应用服务，启动器中可以打开应用、发送到桌面、设置开机自启动或者卸载应用。

龙芯深度应用商店也可以使用 Deepin ID 进行登录，登录后可以对应用进行评分与评论，也可以看到其他用户的评分和评论，是判断应用质量的可靠依据。



背景

2017 年 2 月深度科技与龙芯中科签署了战略合作协议，双方一致认为要继续坚持走自主发展的道路，在充分与上下游合作伙伴进行适配的同时，建立完整的基于“龙芯 + 深度”的生态链。

从 2015 年起，深度科技就已经开始将操作系统移植到龙芯平台上，将近两年的时间，深度科技从内核优化、底层代码逻辑、驱动支持、应用开发、交互设计等多方面逐步完善对龙芯平台的支持和适配，为数个项目打造了良好的基础。

深度科技和龙芯中科一直坚持建立自主可控的国产化软硬件生态环境。上图是最新型号的基于龙芯 3A3000 CPU 打造的全新笔记本电脑。

其他

目前龙芯版深度商店 V1.0 仅适用于龙芯版深度操作系统；如果您发现深度商店中没有的应用，欢迎联系我们（http://bbs.deepin.org）。

想要体验龙芯版深度应用商店，可直接下载龙芯版深度操作系统的镜像，镜像中已经预置了深度应用商店。

龙芯镜像下载地址：http://rsync.deepin.com/deepin-cd/loongson/

深度科技参加国产 CPU 网络安全应用研讨会

深度科技 / 文

由中国计算机学会主办，无锡分布承办的国产 CPU 网络安全应用研讨会，在无锡市信电局的指导下，于 2017 年 3 月 3 日在无锡召开。

本次活动重点研讨基于自主可控 CPU 在网络安全领域的应用现状，以及基于自主可控 CPU 在网络安全领域实现创新超越的可行性和途径。

会议的主题包括：

- SW CPU 发展规划及安全特征
- SW CPU 安全实践
- 基于国产 CPU 的板级设计实践
- 基于国产 CPU 基础软件实践
- 推动基于国产 CPU 安全产业发展

在本次会议中，深度科技总经理刘文欢做了《基于国产 CPU 基础软件实践》的主题发言，发言中重点提到深度科技要实现“真正好用的国产操作系统”的目标：在党政军和关键行业代替微软等国外产品， 实现中国基础软件平台的自主可控。

每一个在成功道路上披荆斩棘的团队，都拥有着为理想一往无前的狂热基因。如果未来能够抵达终点，绝对不是因为对利益的追求和索取，而是因为一身鲜红追逐年少轻狂时想要改变世界的梦。

>>>

因梦而生 大步前行

深度科技售前工程师 王明栋

每一个在成功道路上披荆斩棘的团队，都拥有着为理想一往无前的狂热基因。如果未来能够抵达终点，绝对不是因为对利益的追求和索取，而是因为一身鲜红追逐年少轻狂时想要改变世界的梦……

从零到一

此深度非彼深度

2004 年到 2008 年，深度操作系统是以纯社区化的方式来进行合作开发。主要的产品方向是研发相对于当时网络上其他 Linux 发行版来说，轻量级的 Linux 发行版。一方面体积减少，使其更容易被制作成 U 盘或光盘启动；一方面增加便于使用的 LiveCD 功能；另一方面降低资源占用，以及进行一些本地化工作。对操作系统这几个方面的要求，在现这个用户体验相对友好的时代似乎是稀松平常，没什么特色，但在当时算是一个从无到有的过程和跨越。在当前将操作系统本身做的体积更合适、更易用和更高效的，是一个量的改变，是专业化程度的加深(我们当时还没有能力造更好的轮子，也还没有能力身体力行地促进整个 Linux 桌面应用的生态)，而站在 2004 年这个时间点上，我们则是看到了 Linux 操作系统如果要走向普通用户，这些则是是需要解决的痛点之中比较重要的一部分。

我们不能否认在 2008 年到 2010 年 这一时间段，甚至更早之前，微软的 WIndows 系列操作系统在桌面终端领域取得的傲人成就，这得益于时代的发展、生态的培养，以及人们对于生产力进一步提高的要求。这让我们看到了对桌面终端操作系统的更多需求，让我们更加坚定自己简洁、美观和易用的产品理念。在这一段时间里，深度操作系统团队也由纯社区化运作的方式，逐渐演变为社区化与线下团队相结合的版本推进方式。一方面继续对操作系统本身的体积、易用性和本地化进行深入，另一方面我们开始向深度系统 + 深度精品应用的组合延伸，开始有了自己做原生应用程序的计划。

2011 年初，深度作为一个社区化的公益项目，以工作室的形式在线上和线下社区活跃 (这个时候我们已经能够造更好的轮子，并且能够身体力行的促进整个 Linux 桌面应用的生态)。

当时正在准备版本发布，哪个版本我已经记不得了。设计、网站开发、行政的同学已经下班了。留下来的同学，对于版本发布都是一个萝卜一个坑，一个给 Gnome3 写插件的、一个写软件中心的、一个写深度音乐的、一个写发行注记的、两个代码合并和软件包生成、一个审核软件包和发行注记并合成 ISO 发布。依稀记得我应该是当时连续奋战了 2 天（话说一般版本发布的前一天，都是干通宵的，我可能不止干了一个通宵），在发布的当天，上传错了 ISO，即便很快我就发现 MD5 不对，重新上传了镜像，但还是被熬夜等着我们发布的爱好者发现了。

操作系统处女作

桌面操作系统是否好用，一方面取决于操作系统本身对于技术的透明化程度以及对 UI 和 UE 的优化，另一方面，还取决于在这个操作系统平台之上，用户是否能够通过该平台达到工作、娱乐、学习等方面的需求。所以，深度操作系统由原先集成 Xfce 和 LXDE 桌面环境迁移到了 Gnome3，并首次面向社区发布了应用软件商店、深度音乐、深度影音等原生精品应用。

在国内厂商还在使用 Gnome2，国外厂商使用原版 Gnome3 的时候，我们已经实现了对 Gnome3 的高度定制化，为 Gnome3 贡献了大量的代码和插件。

其实在开源世界，存在很多又好用且功能又强大的软件，完全能够代替 Windows 平台软件的大部分功能。不过这些软件不像 Windows 生态中的应用软件那么容易被大众知道和了解，也甚少有人会将其中的精品软件介绍出来，就更不要提在安装的时候可能还存在技术门槛。基于这些，我们首先开发了原生的应用软件商店。大家找不到软件，我们找好了放进来，大家可能有语言障碍，我们把软件介绍和软件截图都本地化，大家可能有技术门槛，我们把命令行操作变成基于鼠标的界面操作等等，为的也正是将开源世界优秀的软件介绍给大家。

2011 年，考虑到之前工作室的方式已经不太合适了，我们成立了公司，有了一个商业主体。也就是在这一年，才算是发布了深度科技的第一个深度操作系统版本。

羽翼渐丰

一个不用写代码的研发团队

深度系统中软件源就本质来说，其实就是一个软件仓库或者软件集合，在这个仓库中包含了深度系统中各种软件，包括内核、操作系统基础软件、各种程序语言支持以及各种应用软件。软件仓库的质量直接决定了系统本身和应用程序的稳定性和多样化。所以，深度系统研发团队对软件仓库维护的质量，直接决定了整个操作系统的稳定性和多样性。

那时候我们经常通宵写代码，住的近的同学会先回家洗澡、吃饭、换衣服，住的远的同学就只能楼下随便买点吃的，然后在某个时候你抬起头来环视周围，会发现大家怎么又都回来了。我算是住的比较近的，需要先回家洗澡、喂猫、喂狗，然后再回公司继续战斗。通宵写代码的时候，偶尔会有同学在去卫生间的途中在白板上涂鸦，被我随手拍了下来。

（1）深度系统研发团队在维护什么?

软件包一般包括实现一系列命令或特殊功能所必须的所有文件，有两种类型的深度软件包，二进制包和源码包，每个软件包都有一个维护者指定的优先级，用于包管理系统。这些优先级是：

・必须的 (Required)：系统运转所必须的软件包。包括修复系统缺陷所必须的所有工具，不能删除这些软件包，否则系统可能会崩溃，且甚至有可能无法用 dpkg 恢复。仅有这类包的系统是不可用的，但是它为系统管理员启动系统安装其它软件提供足够的功能。

・重要的 (Important)：在任何类 Unix 系统上均安装有该级别软件包。

没有这类包，其它的包无法在系统上正常运转或使用， Emacs 、X11、TeX 等大型应用程序不在此列。此类包构成基本系统。

・一般的（Standard）：Linux 系统里的一般软件包， 构成小型字符系统。

这是用户什么也不选也会默认安装的软件包。不包括大型软件， 但是 Emacs（与其说它是一个应用软件，不如说它是基础构件）一小部分 TeX 和 LaTeX（不支持 X）除外。

・可选的 (Optional)：这包括 X11，所有的 TeX 和许多应用程序。

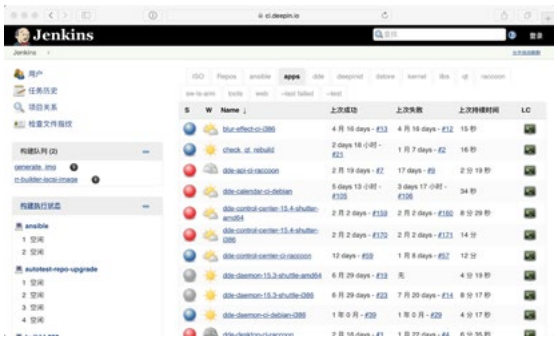
・额外的 (Extra)：这类包不是与其它高优先级的软件冲突，只有知道它的用途才可能对你有用，就是因为特别的原因而不能进入“可选”优先级。

（2）有没有更便捷的技术?

除了这种 Linux 平台传统的软件包之外，系统研发团队还在探索全新的程序包格式。自包容依赖软件包将可以完全独立于系统，它包含了可以运行的所有需要的库及 runtime，并且可通过网络进行更新，同时也可以退回到上一个版本，而不影响系统其他部分的运行。它是一个具有沙箱属性受限的，不可以随意访问外部资源，并和系统其他部分进行隔离。未来自包容依赖软件包的顺利部署，一方面能够大大降低软件包相互依赖所产生的软件冲突，使每个软件都能够依赖最优的库。另一方面，对于用户的安装和升级也会更加便捷，软件包的升级和安装，只限于软件包本身，不再影响系统本身。

一个只写生产力工具的研发团队

在引入持续集成之前，软件开发团队和系统开发团队的日常工作中，有相当一部分精力被用在集成阶段。在这个阶段中，每个独立的开发者或者每个团队将他们开发的各个模块集成为一个可以工作的程序、软件包，进而是整个操作系统。同一个团队成员直接的相互集成与不同团队之间的相互集成，往往消耗了很多战斗力。在深度团队，我们经历了没有构建服务器、晚上进行半自动化构建、晚上进行自动化构建、晚上进行自动化构建和自动化测试、代码质量度量、验收测试和更加自动化的部署，以及持续部署几个大的阶段。在这个过程中，这个专门研发工具的团队从手动集成、手动打包、手动生成 ISO，到自动实现持



续集成，付出了非常大的努力。

·没有构建服务器

软件在开发者的机器上通过手动构建，代码保存在源码仓库中，但是开发者不是经常提交本地的修改。每次需要发布的时候，开发者手动合并修改，这个过程是相当痛苦的。

·晚上进行半自动化构建

在这个阶段，团队有构建服务器，开发者需要手动在晚上执行构建程序。构建过程只是简单的编译代码，没有可靠的和可重复的单元测试。然而，开发人员每天提交代码。如果某个开发人员提的代码和其他人的代码冲突的话，构建可能会失败。

·晚上进行自动化构建

无论什么时候版本管理系统中的代码改变了都会触发编译构建过程，团队成员可以看到是代码中的什么改变触发了这个构建。

·晚上进行自动化构建和自动化测试

无论什么时候版本管理系统中的代码改变了都会触发编译构建过程，团队成员可以看到是代码中的什么改变触发了这个构建。并且，构建脚本会编译应用并且会执行一系列的单元测试或集成测试。除了邮件，构建服务器还可以通过其他方式通知团队成员，如：BearyChat，失败的构建被快速的修复。

·代码质量度量

自动化的代码质量和测试覆盖率的度量手段有助于评价代码的质量和测试的有效性。代码质量的构建会产生 API 文档。

·验收测试和更加自动化的部署

应用不仅仅是简单地编译和测试，而是如果测试成功会被自动的部署到一个应用服务器上来进行更多的综合的 end-to-end 测试和性能测试。

·持续部署

对自动化的单元测试，集成测试和验收测试的信心使得我们可以使用自动化的部署技术将软件直接部署到生产环境中。但是测试还是有可能不能真正的反映现实的环境。

社区

什么是社区?

社区齐聚了一批有共同兴趣的人。一方面在社区中用户与开发者可以进行互动，另一方面会有成员愿意更积极地参与，例如报告 bug、帮助其他用户、撰写文档或进行推广。当然最活跃的社区成员可以得到额外的项目访问权和管理权。

虽然闭源项目也有奖励，但通过奖励社区成员而取得项目贡献毕竟有明确限制。由于不能公开验证代码，因此用户无法真正深入项目、解决问题、开发新功能或者贡献代码。相反在深度社区中，信息（代码和文档）可以从任何社区成员流入社区中心，尽管我们可能会加以调整。更重要的是，一旦出现任何问题，社区中会有很多“眼球”盯着它，以便集思广益、群策群力。

深度社区

深度（http://www.deepin.org）社区是深度操作系统的社区网站，由武汉深之度科技有限公司（以下简称“深度科技”）发起、建立并提供服务和运

深度社区建立了全国范围的志愿者装机协助平台，在安装系统的过程中，遇到远程协助并不能解决所有问题时，志愿者将会在居住点 10 公里之内，提供帮助。

营所需要的资源。专注于 Linux 和深度操作系统在全球的应用和传播，与其他社区相比，它是只专注于深度操作系统的应用、互动和传播，推进深度操作系统在全球的普及和应用的社区，在这里不仅仅有线上的互动，更有线下丰富的互动和开源普及活动。

深度操作系统在很大程度上，是由深度科技支持开发的。社区是深度操作系统开发、发布和维护不可分割和极为重要的组成部分。深度社区拥有会员、贡献成员和正式成员以及各个专业组的成员。深度社区为全球各地的参与者提供了参与深度操作系统系统开发、发布和维护的平台；为用户提供了沟通、交流和学习的平台，也为 Linux 爱好者提供了更多参与开源项目的方式方法。沟通交流和参与开源项目在深度社区中变得简单、高效。

发展路线

开始的时候深度社区还是比较小的，只有我们自己团队的开发者，几乎没有用户。不过社区一般都会有一个“孵化期”，在此“孵化期”，我们作为初始团队会努力工作打下基础。“早发布，多发布”是一条众所周知的开源开发准则，因为这样做可以获得宝贵的早期反馈，帮助项目团队建立自信。另外，在管理好期望值的基础上，在社区积极发布版本也使得我们受益良多。

产品要最终吸引用户，产品的演示和品牌推广必须让潜在用户相信，该产品与竞争性产品相比有显著优势。引起用户的兴趣后，必须要降低进入的门槛：例如我们降低了系统安装和使用的门槛，当然，吸引更多用户注册并不是这个故事的结尾。我们同时还积极与开发者沟通，帮助开发者更容易的参与进来。

成长中的深度社区

从一开始，我们就明白围绕开源操作系统建立社区是一项缓慢而艰巨的工作，但同时，我们也明白建立一个以深度操作系统或 Linux 操作系统为主

题的社区的重要性。之前也有一些社区为深度操作系统设立了专区，但我们依然决定在 2011 年着手建立属于深度操作系统自己的社区。从最开始的论坛、到 wiki、到博客、到全球镜像站点，到现在一次次对表现形式的更新和优化，社区的存在和繁荣更好的支撑了项目本身。但所有社区都始于被产品的包装、品牌推广或口碑推荐所吸引用户，只有不断满足他们对产品和团队不断提高的期望值，才能够将我们对于开源与操作系统的理念和执着，传递的更加深远和广博。

现在，越来越多的用户加入到深度论坛中来，他们来自社会各个阶层，来自不同的国家，但都是基于对深度操作系统的认可而来，他们的发帖鼓励并帮助深度科技做出更好的操作系统。

参与国家信息安全的新开始

国产操作系统的现状

针对桌面客户端操作系统市场和服务器端操作系统的市场而言，目前中国拥有巨大的市场容量。根据统计，目前中国桌面终端设备存量为 3 亿台，其中党政军等涉及国家安全的行业使用桌面终端超过 3000 万台。按照每年 20% 的更新速度和桌面操作系统平均价格估算，桌面操作系统的年度市场份额约为 100 亿元人民币。但这个市场基本被微软的 Windows 系统垄断，全球市场基于 Linux 的桌面操作系统约 1.78%。在中国市场，基于 Linux 平台的操作系统长期没有超过 0.2%，相比之下 MacOS 已经在中国市场使用率已经超过 1%，微软的市场占有率超过 97%。根据国家版权局的数据，中国的软件盗版率不超过 50%，也就是说每年在桌面操作系统市场，微软会拿走 50 亿以上的授权费用。

中国服务器市场规模也相当庞大，2015 年仅 X86 服务器出货量已经超过 200 万台，按照服务器操作系统平均价格 5000 元估算，每年的服务器市场规模也超过 100 亿人民币。其中 Linux 服务器应用范围超过 30%，国产操作系统厂商的潜在市场超过 30 个亿。在国产服务器操作系统方面，尤其是特定行业与政府机关，国产操作系统厂商相对成功，也做出了必要的贡献。但从行业内调查统计得到的数字来看，国产操作系统和国外操作系统（redhat+SUSE）的销售额总数仅为 3~5 亿人民币，并且其中有大量服务器并未从原厂购买商业 License 和 service。

我们认为，当前 Linux 操作系统的应用在全球范围内正在呈上升趋势。随着 Linux 服务器操作系统日趋成熟，越来越多的在社会关键领域承担重要角色，在移动终端领域，基于 Android 的终端系统也进一步高歌猛进。Linux 操作系统已经证明了自身具有广泛引用的基础。在这个关键的时间点，深度科技经过多年的努力，集中优势力量大力研发具有自主知识产权、具有自身特色、简单美观易用、能够满足日常办公应用，且可用于替换 Windows 操作系统的深度操作系统，就时间特性来说，具有时代竞争优势。现在，中国这个时代正在呼唤等同于 Windows 桌面操作系统的出现，而深度操作系统则恰逢其时的成熟于这个时代。

深度操作系统所具备的优势

操作系统从应用范围大致可以分为终端操作系统（包括 pc，有屏幕交互能力移动终端等），服务器操作系统（用于互联网服务器，企业服务器等中心服务设备上的操作系统），工业控制操作系统（工业控制设备，网络设备较多使用，大多数是类 Unix 的实时操作系统），智能设备操作系统（无屏幕，应用在消费市场的智能设备，大多数基于嵌入式 Linux）。操作系统以不同的形式深入在社会生活的各个角落。

深度科技及产品的优势所在：

深度科技致力于发行
美观易用，安全可靠的国产操作系统产品

1、做为纯粹的民营企业，深度科技具有自身的生存能力。

国产操作系统厂商过度依赖政府扶持和课题资金注入，前期政府推动有必要，但是过于依赖政府项目，生存无忧后出现了目标和理想的缺失。

深度科技从最早的社区起步，因理想而出生、为责任而成长，创造优秀的自主可控操作系统，并通过产品能力占领市场，因而取得市场地位，是深度科技的目标和理想。

2、产品技术创新和产品技术能力，是深度科技生存的根本。

加强技术创新，形成核心‘技术竞争力’是每个国产操作系统厂商必须要做的事情。

深度科技集中国内优秀的操作系统开发精英，深入研发多年，形成了自身独特的技术优势。深度的操作系统研发团队结构，不同于国内和国外其他操作系统厂商，其创新的方向集中在用户体验和功能适用。在桌面操作系统方面，着重在用户体验，在服务器操作系统方面，则注重用户功能适用，系统功能合理，安全策略明晰等多种特点。

3、以服务为先导，是深度给客户始终如一的承诺。

深度科技倡导以服务为先导，将产品和服务一致化。产品是服务的基础，服务是产品的延伸，以向用户输送价值为深度产品和服务的最终目标，获得了众多用户的好评。

4、让市场说话，是时代给予深度的良机，也是深度发展的沃土。

市场对产品的认可会从产品品牌认知到产品价值，再从产品价值到产品品牌再认知不断螺旋提升。

深度科技保持围绕产品技术为基础，通过展开深度多年在社区、产品能力、服务质量多个方面积累，向用户展现产品品牌；进而促使用户使用和感知深度产品和服务的价值，形成产品品牌提升的再认知。逐渐的，形成产品市场地位，得到用户的不断认可。深度科技不断的以这种模式突破市场，巩固优势。

操作系统不是一个或者几个厂商就能支持起来的，而是需要一个完整的生态圈，建设生态圈需要整个行业各个层面的参与。深度认为要形成生态圈的突破口在于实际使用占有率能否超过 5%，一旦过了这条线，市场这只无形的手就能指挥起各家厂商共建生态系统。如果想在短短的几年内达到这个百分点，通过纯市场化的方式，基本不可能实现。但借助”斯诺登”事件的影响，借助“xp 停服”的空窗期，首先在政府机关实现国产操作系统替代，年替代率可能能达到数百万台，这已经超过预期国内出厂占有率的 5%，是完全有机会达到这个量级的。

依法强化关键信息基础设施安全保护

宁家骏 南风一号 / 文

人类已经进入网络信息时代这样一个历史阶段，这是一个世界潮流，而且这个网络信息时代对人类的生活、生产、生产力的发展都具有很大的进步推动作用。网络信息技术全面融入社会各个领域，不但给人们的生产、生活等带来越来越深刻的变化，而且已经直接影响到国家、区域和全球竞争力，也将维护网络安全、建设网络强国上升到国家战略的高度。当前的社会已经是信息社会，网络不仅是基础平台，也已经成为经济社会的内在组成部分，且信息网络在经济和社会发展中的作用越来越突出，加快从“网络大国”向“网络强国”转变，成为我国既定的战略目标。

当前，金融、能源、电力、通信、交通等领域的关键信息基础设施是经济社会运行的神经中枢，是网络安全的重中之重，是网络战首当其冲的攻击目标。党的十八大以来，习近平总书记多次在不同的场合提出了建设网络强国的目标。特别是 2016 年 4 月 19 日，习近平总书记在网络安全和信息化工作座谈会上明确要求“树立正确的网络安全观，加快构建关键信息基础设施安全保障体系”。

一、进一步提高对加强关键信息基础设施保护紧迫性的认识

国家关键信息基础设施是网络安全的重中之重。能源、电力、通信、交通等领域的关键信息基础设施是经济社会运行的神经中枢，也是可能遭到重点攻击的目标。攻击后果从信息窃取发展到系统和设备毁伤，乃至可能造成人员伤亡，呈现出破坏性不断增强的趋势。关键信息基础设施中防护手段建设成本更高、更新维护难度更大，而攻击成本仍十分低廉，呈现出更加非对称的特征。

经过近年来有关单位的监测，目前我国重要信息基础设施依然普遍存在着安全漏洞，而且这些漏洞存在被利用的巨大风险，利用这些漏洞可轻易获取系统控制权限，涵盖制造、电力、交通、水利、燃气、智慧城市等多个领域共 30 多个系统。监测还表明，对于重要信息基础设施而言，来自境外的安全威胁日益严重，包括境外机构对其进行扫描探测和进行渗透攻击。国内专业机构监测还表明，这种网络安全攻防对抗的发展趋势将有增无减，进一步加剧。第一，针对国家关键信息基础设施的网络攻防对抗研究将成为各国政府、安全界乃至暴恐组织的关注重点，相关的安全事件将持续增加。第二，对绝大多数关键信息基础设施而言，无论联网与否，都不是保证其安全的充分条件。特别是随着互联网与信息技术的应用不断拓展，开放互联是大势所趋，它们也因此将面临更为严峻的网络安全威胁。第三，对绝大多数关键信息基础设施而言，都有诸多必须使用的产品处于信息化与自动化的结合环节，它们因此也成为网络攻击的主要对象和核心环节。针对这种严峻的形势，必须尽快建立健全安全威胁情报系统与应急处置体系，加强网络安全知识共享与行业的深度融合合作。

二、《网络安全法》通过是我国关键信息基础设施保护方面的里程碑事件，具有重要意义

日前通过的《网络安全法》用大量篇幅规定了关键信息基础设施保护相关内容，进一步界定了关键信息基础设施范围，同时对攻击、破坏我国关键信息基础设施的境外组织和个人规定相应的惩治措施等，国家关键信息基础设施网络安全迎来了新发展。

特别值得注意的是，在我国首部网络安全大法中，已将关键信息基础设施安全保护制度确立为国家网络空间基本制度，关键信息基础设施安全保护办法将由国务院制定。《网络安全法》第三章专门对关键信息基础设施的运行安全提出了具体要求。这体现了我国对关键信息基础设施安全的高度重视，表明我国对关键信息基础设施安全保护上升至前所未有的高度。

这部法律的颁布和实施意义重大。首先是关键信息基础设施的定义和边界进一步加以明确。2003 年我国就在相关文件中首次提出“重点保障基础信息网络和重要信息系统安全”，并通过实践明确了基础信息网络是广电网、电信网、互联网，重要信息系统是银行、证券、保险、民航、铁路、电力、海关、税务等行业的系统，即俗称的“8+2”。为了进一步明晰我国关键信息基础设施范围，《网络安全法》中首次明确了关键信息基础设施的原则性范围，规定“国家对公共通信和信息服务、能源、交通、水利、金融、公共服务、电子政务等重要行业和领域，以及其他一旦遭到破坏、丧失功能或者数据泄露，可能严重危害国家安全、国计民生、公共利益的关键信息基础设施，在网络安全等级保护制度的基础上，实行重点保护。”这是我国首次在法律层面提出关键信息基础设施的概念，明确关键信息基础设施涉及的主要行业和领域，为我国明确关键信息基础设施的定义范畴提供了法律依据，是开展关键信息基础设施安全保护的基础。

其次，鉴于关键信息基础设施保护涉及不同行业、领域，形势错综复杂，要求高、任务重。因此对关键信息基础设施进行安全保护，不仅需要提高关键信息基础设施自身安全，更需要进行一系列制度规范体系建设，构建多边的协调机制，建立完善的规章制度。对此《网络安全法》提出了建立健全网络安全监测预警和信息通报制度、建立网络安全应急工作机制、制定应急预案和建立关键信息基础设施运营者采购网络产品、服务安全审查等一系列重要制度的要求，为关键信息基础设施安全保护搭建了制度框架。这部《网络安全法》的出台将促进配套法规办法的制定与完善，特别是促进相关行业关键信息基础设施保护方面的制度建设，对上承接网络安全法，将本行业关键信息基础设施提出的安全保护要求落地。

三是为了有效开展关键信息基础设施保护工作，《网络安全法》明确了不同主体的安全责任以及相互关系，为关键信息基础设施安全保护规定了责任划分和追责方式。《网络安全法》从国家、行业、运营者三个层面，分别规定了国家职能部门、行业主管部门及运营企业等各相关方在关键信息基础设施安全保护方面的责任与义务，从法律层面约束有关方面加快落实和推动责权清晰的管理体系建设。

第四《网络安全法》进一步强化了我国对关键信息基础设施的网络安全管辖权，确立了我国对关键信息基础设施不可侵犯的网络主权。明确规定了所有境外的个人或者组织从事攻击、侵入、干扰、破坏等危害中华人民共和国的关键信息基础设施的活动，造成严重后果的，必须依法追究法律责任；国务院公安部门和有关部门并可以决定对该个人或者组织采取冻结财产或者其他必要的制裁措施。这就为相关部门打击境外对我关键信息基础设施的网络攻击等行为提供了法律支撑。

《网络安全法》通过是我国关键信息基础设施

保护方面的里程碑事件，具有重要意义。目前，当务之急是要切实做好《网络安全法》的贯彻实施，有效运用这一强有力的法律依据，为我国关键信息基础设施网络安全保护工作的顺利实施保驾护航。

三、以自主创新引领，切实增强我国关键信息基础设施保护能力

2016 年 10 月 9 日，中央政治局进行第三十六次集体学习，主题是实施网络强国战略。习近平总书记在主持学习时强调，加快推进网络信息技术自主创新，加快数字经济对经济发展的推动，加快提高网络管理水平，加快增强网络空间安全防御能力，加快用网络信息技术推进社会治理，加快提升我国对网络空间的国际话语权和规则制定权，朝着建设网络强国目标不懈努力。这是对我们进一步做好重要信息基础设施网络信息安全工作的又一次高瞻远瞩的理念诠释和最及时的重大工作的及时部署。

为了建设网络强国，切实做好关键信息基础设施保护，最重要的路径设计就是顺应网信事业发展大势——加快核心技术自主创新。实施网络强国战略，确保我国关键信息基础设施网络信息安全，技术可谓重中之重。当今世界网络信息技术研发投入最集中、创新最活跃、应用最广泛、辐射带动作用最大，技术革命一浪赶一浪，做到顺势而又不是被动地跟随潮流，核心技术的自主创新就成了最关键的所在。总书记多次讲话都特别强调“核心技术受制于人是我们最大的隐患”，不能只“在别人的墙基上砌房子”，一定要紧紧牵住这个“牛鼻子”，解决我们发展中遇到的最大“命门”。核心技术是国之重器，最关键最核心的技术要立足自主创新、自立自强。市场换不来核心技术，有钱也买不来核心技术，必须靠自己研发、自己发展。特别是在国家关键信息基础设施中一定要加快推进国产自主可控替代计划、构建安全可控的信息技术体系、实施网络信息领域核心技术设备攻坚战略等举措，尽快超越“跟跑”阶段，争取在某些领域、某些方面实现“弯道超车”，才能保障国家关键信息基础设施网络信息安全、国家安全。

坚持自主创新，加快推进国产自主可控替代，是实施网络强国战略，审时度势、固本强基、顺势而为的智慧之举，也是掌握我国关键信息基础设施安全保障的主动权的明智之举，必须依靠政府、社会以及各行各业的共同努力。要在关键信息基础设施建设和发展中进一步增强网络主权意识，下大力气研发过硬的技术，政府应该做网络技术的坚强后盾和重要保障，勇于担当，做关键信息基础设施安全保障体系建设的引领者。

建设健全关键信息基础设施安全保障体系离不开各行各业的协同能力，企业应该成为建设关键信息基础设施安全保障体系的主力军，在这一工作中担负着重要的历史使命，是关键信息基础设施安全保障的力量之源、技术之源，是技术创新、网络安全的重要推动着，也是关键信息基础设施安全保障体系的用户责任单位和维护者，必须肩负起主人翁使命，增强主动性、能动性，为促进网络健康发展、捍卫网络空间安全作出应有贡献。进一步增强我国关键信息基础设施网络信息安全保障能力。并逐步形成完善的保障体系，是建设网络强国的核心使命，也是实现中华民族伟大复兴梦想的重要支撑，我们必须尽一切努力撑起这面大旗！

我们一定要深入学习领会习近平总书记重要讲话精神，把国家关键信息基础设施网络信息安全保障工作结合起来认真贯彻落实，强化安全，细化服务，切实提高水平，发现人才，培养队伍，不断完善服务质量，强化安全防范，确保安全。

学习习总书记网络安全观，就是要正确、全面地把握习总书记阐述的网络安全观的理念，科学设计国家关键信息基础设施安全保护的相关战略与发展路径，使其成为实现网络强国的真正驱动力，达成普遍共识，我们才能战略清晰、应对得当，走出网络大国到网络强国的创新发展的康庄大道。

倪光南院士：
中国为何做不出像样的操作系统

科学网 / 文

放眼世界，当前哪个行业最垄断？在 2016 年 12 月下旬举行的 2016 中国大数据大会上，中国工程院院士倪光南给出了这个问题的答案：智能终端操作系统。

有人觉得航空飞机的垄断性最强，倪光南不以为然：“航空飞机被波音、空客所垄断，总数量也可能只是数十万级别。但全世界几十亿台智能终端只有三种操作系统：苹果、安卓和 Windows，这种垄断在全世界找不到第二例。”

终端操作系统受垄断

倪光南指出，智能终端是产生大数据的重要来源，多种形式的大数据即是通过终端产业而来。同时作为接受大数据云服务的主要载体，这些终端的安全在很大程度上决定了大数据的安全。“（智能）终端的安全与大数据安全关系密切。”

他认为，操作系统可以轻易控制电脑、手机等终端，是智能终端安全的“总阀”。“智能终端操作系统的垄断不打破，终端安全和大数据安全也就无从谈起。”

也正因此，倪光南提出，中国应该坚持移动信息化核心技术自主创新：不仅要防范智能手机操作系统的安全隐患，更应吸取教训，培育和使用自己的操作系统。

“从网络安全角度，中国要成为网络强国，必须解决智能终端操作系统被垄断的问题。”倪光南说。

然而，中国在信息核心技术领域，尤其是 CPU 和操作系统这两方面的弱势很明显。美国能够引领信息领域，重要原因便是掌握了 CPU、操作系统这两项核心技术，它们堪称是 IT 皇冠上的明珠。

“操作系统在某种程度上影响更大，世界上最大的三家 IT 公司是苹果、谷歌和微软，一个做苹果系统，一个做安卓系统，一个做 Windows 系统，应该说这不是巧合。”倪光南说。

国产操作系统待突围

近年来，中国在操作系统领域取得了一定的成绩，也研发出了一些国产操作系统。但目前国产操作系统装机量仍然很少，这就造成了相应的应用开发“没有鸡生蛋，就没有蛋生鸡”的问题——没有应用，就没人愿意用国产操作系统；没人用操作系统，就更没有人愿意开发应用。因此，国产操作系统的处境仍是非常尴尬。

为什么中国做不出像样的操作系统？倪光南说：“简言之，我们总体实力比美国这样的领头羊差，而中国像华为这样肯在研发上下真功夫的公司还很少。”

此外，倪光南还指出，中国最大的问题是“各自为政，合作意识很差”。

“中国做终端操作系统的有十多家，这肯定是太多了。”倪光南认为，这样会有太多的内耗，结果就是难以成事。“我们的终端操作系统出不来，资源分散是很致命的。”

在一次接受媒体采访时，倪光南就抛出这样的看法。他说：“中国科技人员的最大毛病，是宁做鸡头，不做凤尾；三个和尚没水喝，就是团队合作精神较差。在信息领域，这个缺点也存在。当然，中国有很优秀的个体，比哪个国家都不差。因此，我们应当扬长避短，充分发挥中国的人才优势。”

倪光南常以北斗卫星导航系统的成功作比：“当初美国卫星导航系统（GPS）研制出来，民用是 10 米级精度，军用是 1 米级精度，似乎大家都可以用。但是，GPS 那么好用却不能依赖它——实际上没有 GPS 会挨打，用了 GPS 可能挨打更严重。这样，中国被迫发展出了北斗系统。”

中国机遇尚存

在安卓系统的基础上做终端操作系统，行不行得通？倪光南认为“不行”：“安卓系统说到底还是别人控制的，大部分是开源的，也有一部分不开源，但是开源以后控制权就不在你这了。”

倪光南认为，游戏规则决定了中国不能依赖安卓操作系统，“我们要发展自主可控的、本地化或者定制化操作系统，还是要考虑自主创新”。

从 Windows Phone 的发展情况来看，在 iOS 和安卓生态已经形成垄断的情况下，新的移动操作系统很难发展起来。中国还有没有可能再做出新的操作系统？

倪光南的答案是肯定的：“我们不满足于做网络大国，中国的目标是做网络强国。此外我们有安全需要，一些行业非常愿意做自己特殊的系统，这是一方面；另外现在软硬件的发展使得当下双系统很流行，不需要付出任何代价就可以用两个系统。一个是常用 OS，一个是安全 OS，随时可以切换，这种方式我认为可以支撑我们的系统。”

倪光南主张中国操作系统要从北斗卫星导航系统的成功中汲取经验：“按说做操作系统不会比做卫星导航系统难——北斗是航天和信息这两个领域技术的融合，需要投火箭、卫星再加上许多硬件和软件，而操作系统基本上只需投入智力就可以了。北斗的成功体现了中国“两弹一星”和载人航天精神，体现了中国集中力量办大事的优势，体现了我们信息和航天领域的实力，体现了我们的综合国力。由此可见，中国的操作系统上不去，应该主要是自身的问题，是没能发挥自己的优势。”

国产操作系统份额难破 5% 关卡 四大推广难点成拦路虎

◉ 通信信息报 林永华 / 文

2016 年，尽管国产操作系统取得了一些发展成果，但发展依旧缓慢，而这一态势在 2017 年或将继续。调研公司 StatCounter 不久前公布的调查数据显示，微软系的操作系统仍然在市场上占据着绝对统治的地位，而国产操作系统占比较少。中国工程院院士倪光南在 2016 年 10 月末表示，国内市场上国产操作系统的份额只有 3% 左右，换而言之经过几年的发展，国产操作系统本土的份额仍然不及 5%。

当前，国家已经确定国产操作系统将作为未来网络强国战略目标之一，在政策上给予了相应扶持，然而历经数年，国产操作系统依旧发展较慢，用户粘性差、企业换平台成本高、国产操作系统用户接受力弱、盗版猖獗四大阻碍悬而未解是根本原因。迈进 2017 年，国产操作系统厂商应努力攻克这四大弱项。

国产操作系统处境堪忧

StatCounter 的调查数据显示，在 2016 年 12 月，中国用户量最大的操作系统仍然是 Windows 7，49.36% 的份额无人可撼动；第二位是 Windows 10 份额为 18.52%，XP 则以 18.36% 微弱差距排在第三的位置；Win8 则占有 1.02%，在几大操作系统占大头的状态下，留给国产操作系统的份额并无多少。中国工程院院士倪光南 2016 年 10 月末接受媒体采访时表示，国内市场上国产操作系统的份额只有 3% 左右，而这也意味着在经过几年的“大推进”下，国产操作系统在国内的份额仍然不及 5%。

2013 年斯诺登棱镜门事件的爆出，以及微软停止对 WinXP 所有版本支持，引发了国家对于网络安全和国家安全的重视，没有网络安全就没有国家安全已经成为业内共识，国产操作系统大发展也由此开启。在 2015 年操作系统市场中科系、中标系、新支点三足鼎立的局面逐渐形成。除此之外，deepin linux、普华 Linux、Lomo Linux、威科乐恩 Linux 等国产操作系统不断发力，力争市场上游。

然而看是一篇火热的市场背后，三年的发展结果却让少不了伤感。

四大难关仍未跨越

当下的国产操作系统处境十分尴尬，而造成“努力发展却无法百尺竿头更进一步”的原因主要有四个方面。

首先，用户对微软操作系统的黏性制约国产操作系统推广。当前国内绝大多数的用户使用的

均是微软的操作系统，在习惯了该操作系统之后，要让用户就同一个操作系统换一个版本的难度都十分大，从此前微软升级 Windows，用户的反映中便能略窥一二，更何况是让用户彻底的换操作系统。

其次，在企业级市场上的推广受到企业更换平台成本较大因素限制。不少企业发展过程中积累了大量的数据和资料以及应用系统，想要这些企业使用国产操作系统，就必须将数据和资料以及应用转移到新平台，而这产生的成本并不小，加之还需要冒着数据丢失与损毁的风险，企业自然对国产操作系统的兴趣了了。

再次，预装国产操作系统阻碍大。此前一些国产操作系统厂商在遇到推广困难时，便希望借鉴智能终端预装 APP 的方式将国产操作系统预装在终端设备上。然而，现实总是很残酷，不少销售商在安装国产操作系统的设备上安装 Windows 售出，让这些国产厂商的算盘落空。

最后，盗版问题也是阻碍国产操作系统的发展的一个重要因素。当下市场中仍存在盗版的 Windows，几乎零成本的安装费用让用户在选择操作系统时，根本就无需考虑，这也造成了一个结果，即使国产操作系统免费，依旧少有人问津。

前路漫漫，仍需不断攻克难关

回顾 2016 年，国产操作系统虽然整体的市场份额并未突破 5%，不过仍需看到国产操作系统在进步，尤其是在政策的支持下，国产操作系统在政府、国防、金融、教育、财税、交通、医疗、制造等方面已经有了突破口，多个领域已经进入核心应用部分，尤其在审计、财税、中国航信、中纪委、工商等领域取得了较强的市场占有位势。但对于国产操作系统长远发展而言，这是不够的，迈入 2017 年，国产操作系统厂商应该更加努力。

一方面，心态仍需摆正。相比国外传统操作系统厂商，国产操作系统厂商无论是在人力还是物力上的投入都有着很大的差距，在产业链还不够完善的前提下，国产操作系统厂商要齐心协力，避免针对某些核心技术的重复投入过多，在某些市场活动中的“相互拆台”的事件发生，应从努力为打造国产信息化建设作贡献的角度出发，共同探索与努力。

另一方面，从用户的角度出发，探索自身的不足。时下国产操作系统的最大问题在于生态圈还远不如微软 Windows 成熟，缺乏 Windows 系统那样丰富的应用，如何吊起开发者的兴趣，丰富产业生态，从而吸引用户，增强用户使用国产操作系统的黏性，都是需要国产操作系统厂商在 2017 年多下功夫。

总之，在信息安全越来越严峻的当下，国产操作系统厂商们需要避免闭门造车，而要通力合作才能在未来操作市场上赢得更多的机会。

深度操作系统 15.4 给你“好看”

>>>

深度操作系统是一个致力于为全球用户提供美观易用、安全可靠的 Linux 发行版。

深度操作系统 15.4 全新设计了控制中心，并默认预置了深度家族的一系列应用，不仅做到好用还要好看，还提升了用户体验。另外，新版自带内核升级到最新稳定版，系统稳定性和兼容性方面得到了大大的提升，全新的深度安装器界面，让安装也非常简单。此外，得益于社区用户热情反馈，深度操作系统 15.4 在功能完整性和稳定性方面得到了显著改善。

新版控制中心　设置如此简单

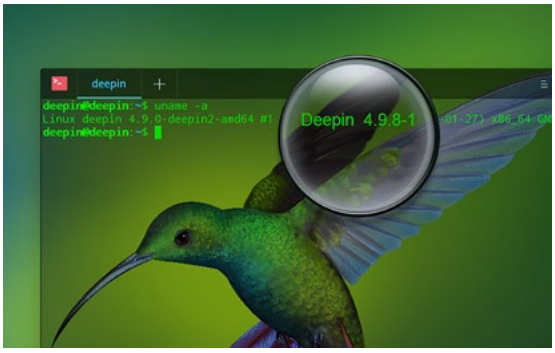
全新重写并设计的控制中心，首页承载了天气插件、通知中心、相关快捷等常用操作，设置变得更加简单明了，同时全新的交互体验和设计让系统更加完美。

全新安装界面　一杯咖啡时间

集成新版深度安装器 2.0，全新的安装界面、智能化的安装检测、贴心的提示交互、二维码模式反馈问题，让安装也变成一种享受。

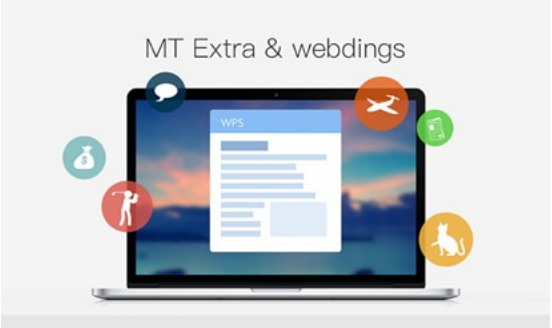
内核跟随主流　兼容更多可能

默认升级内核到 4.9.8 版本，提高系统稳定性和兼容性；集成更多的驱动和硬件支持，满足最新六代 / 七代 CPU 用户的需求。



重绘图形字体　告别弹框提示

重绘并集成 MT Extra 和 webdings 图形字体，从此告别打开 WPS 文件提示字体缺失弹框烦扰，安静的书写每一笔。



美化窗管热区　多指手势触摸

优化窗口管理器交互效果，工作区设置不同壁纸，系统默认集成触摸板多指手势功能，从此告别鼠标。

开源　分享欢乐

深度操作系统是一款针对普通用户而发行的开源桌面系统，您可自由下载、分发、修改和使用。

GitHub：https://github.com/linuxdeepin

欢迎您关注我们的微博、微信（深度操作系统）、Twitter 和 Facebook，以第一时间获取最新动态，同时也欢迎您前往我们的论坛，与我们交流和分享您的快乐。

最后，我们郑重感谢为深度操作系统提供测试、文档、翻译和镜像支持的社区团队与企业，感谢你们的无私的贡献，开源有你们更精彩！

官方下载点：

64 位：点此下载（MD5 值：e679dc41a0220566e763e116fdf4fd3a）

深度操作系统 15.4 版本将不再提供社区版等 32 位 ISO 镜像下载，如需获取或商业支持，请发送邮件至 tech@deepin.com。

其他下载点：

百度云、Google Drive、MEGA

15.4 研发心得 >>>

一群人的狂欢

江湖传言，深度的系统只有在发布新版本的时候是最稳定的，从某种程度上来说，是这样的：每次系统发布前的一个月内，研发团队的各个部门都会像打了鸡血一样，不管是哪个项目组都能集结在一起，撸起袖子不眠不休地干，直到最终版的 ISO 静悄悄地躺在服务器上方为止。发布前的这个月内，tower 上会热闹非凡，各种需求会被反复讨论、修改和重新编码，产品撕设计、设计撕开发、开发奋力改代码，再加上测试的介入……我想如果把这叫做一场战争，似乎并不为过。这是一个团队跟竞争对手的战争，更是一个团队对自己的战争，虽然乱，但是乱中有序；虽然累，但是苦尽甘来。每一次经过这种战争的洗礼，参与者会有进步，团队会有进步，我们的产品也会有质的飞跃。

比较了解我们系统发布周期的人都知道，每个新版本的研发周期大概都定在三个月左右，因为需求是无限的，时间如果也不固定，那么无限的需求意味着需要无限的时间，所以只能通过时间来约束需求，这叫“时间决定需求”理论。这么多次系统发布，可以说每次都有非常不同的经历，例如，V15 是我第一次开始接手系统发布，慌张、手足无措；V15.1 因为 ISO 存在严重问题，不得不紧急发布 V15.1.1；V15.2 中畅快淋漓的新 Launcher 和广受好评的网易云音乐；V15.3 中再也不歪歪扭扭的新 Dock。同样，V15.4 带来的又是完全不一样的体验，说它特别，特别在更新内容之丰富，也特别在其中一些不同寻常的发布经历。

硕果累累

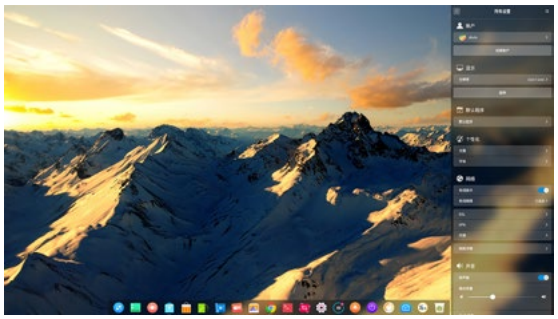
以前参加一次黑客马拉松的时候，有个 IT 爱好者对我说，Mac OS 有一个很吸引人的地方在于，每次系统更新它都会带给人一种全新的感受，包括视觉上和体验上，有时候甚至感觉像是新买了一台笔记本。这句话我想了很久，觉得很赞同。超出用户预期的东西，多少都更容易打动用户。现在，如果你有一台装了 deepin 的电脑，那么这次 V15.4 的升级绝对会让你有这种“像是新买了一台电脑”的体验：

首先是依照用户习惯重新设计的控制中心，快捷操作栏解决了用户常用的设置找不到的问题，一般的使用场景下，用户再也不需要到繁杂的设置中找到需要的连接 WiFi、开启 VPN、连接蓝牙设备等功能；而贴心的小插件让用户可以添加自己喜欢的扩展到自己控制中心的首页，千呼万唤的通知中心功能就是作为默认的插件集成到了首页的小插件中。

所有设置界面更是做了全面改版，从原来完全分隔开来的模块设计改成更现代化的滚动设计，好看亦好用。

多工作区壁纸的支持、切换工作区的表现形式

全新的控制中心 – 首页



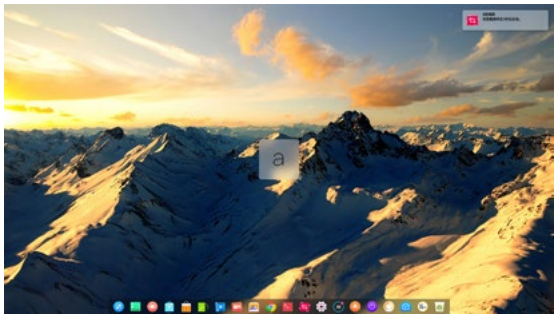
全新的控制中心 — 所有设置

以及终于用好看的壁纸盖住了窗管背后丑陋的单灰色界面等，窗管也迎来了一系列优化和改进。

极致打磨的窗管

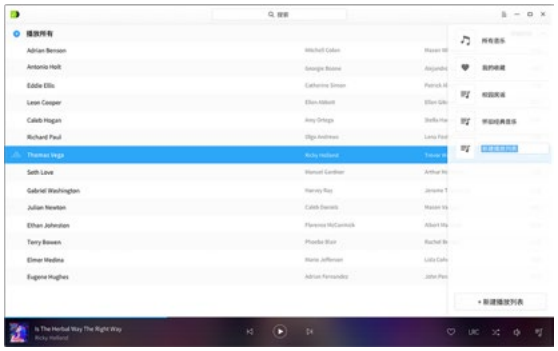
极致打磨的窗管

deepin 作为现代化的桌面操作系统，OSD、菜单、通知和 Dock 都采用了同控制中心一样的毛玻璃效果，有没有感觉系统像羽毛一样轻?



全新的毛玻璃设计

曾经广受好评的深度音乐，终于迎来了新版；全新的设计简洁、大方。



全新的深度音乐

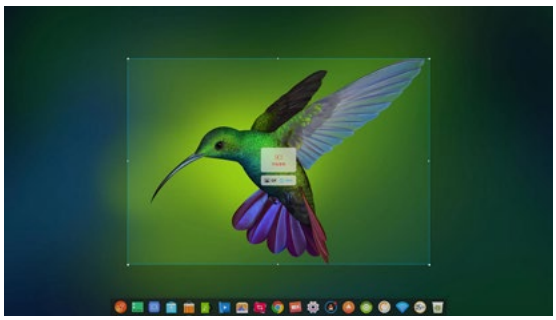
全新的安装器，一杯咖啡的时间，即可享用。

全新的安装器

以及全新设计的 U 盘制作工具，保留原来操作习惯的同时，极大幅度的提高了应用的颜值。

全新的 U 盘制作工具

还有深度家族的新成员——深度录屏，承袭深度家族高颜值的特点，自“出生”就广受好评。虽然目前还是 Beta 版本，但是已经相当稳定。



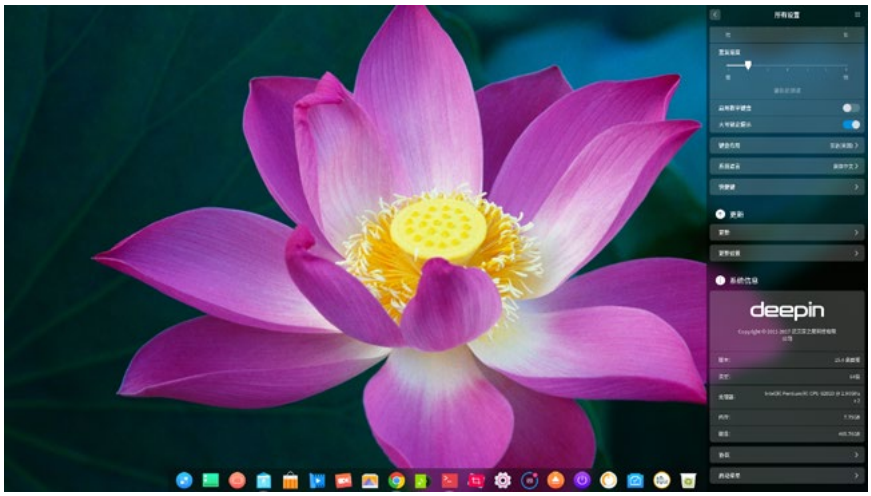
全家桶又添新成员——深度录屏

除此之外，深度家族的文件管理器和看图也将迎来一次大更新……像这种一次版本发布能带着这么多更新和新产品的，15.4 当属第一。不仅如此，一些重新进行架构设计的项目，例如控制中心，在速度和资源占用上的优化也都集成在这个版本中，让你的系统颜值增加的同时，也能享受极致、流畅的体验。

协作

“所有美好事物的背后都有一个辛勤耕耘的团队！”一位伟人说过。上面的累累硕果肯定让人振奋，但是研发团队研发 15.4 的过程却不是一帆风顺的，可以说是痛苦的。很多东西都是经过反复沟通和修改的，每一个细节，动画、线宽、色彩和交互，先是产品团队和设计团队做预先讨论和评审，再给开发团队和测试团队进行需求讲解，一是确保开发团队和测试团队能充分理解每一个需求点；另一个是在这个过程中又会有很多模棱两可或者不准确的需求被发现。就算是这个过程也顺利通过，谁又能确保开发在实现的时候不会发现新的设计缺陷？对于一个在细节上追求完美的团队来说，方案可能被小修小改，也可能被完全推翻。这样反反复复一直到系统发布前的一两个星期才算完……也不是说没得改了，而是三个月期限到了。

曾经在一本书里面看到了两句诗：“独行潭底影，数息树边身”，顿觉找到了人生知己，贾岛他老人家虽然不一定知道操作系统是什么东西，但是我却能深深地理解他这种孤独感。这不是苏轼说得“高处不胜寒”那种孤独感，而是真实的“人好少啊”这种孤独感。这种感觉在 V15.3 时尤甚，因为我们组负责桌面环境，你在 deepin 上每天都会看到的任务栏、启动器和控制中心，以及时不时跳出来的各种对话框、菜单，都是桌面组在维护。还有一些可能比我们工龄还大的“老项目”，没有人维护的时候就需要桌面组接手，而 15.3 的时候桌面组忙得焦头烂额，其他组的各路大神也都忙的不亦乐乎。



事或者其他原因，“需求说啥，我做啥”，这种做法不仅浪费自己的时间，也坑用户。

很多人甚至包括我自己其实都对我们的产品到底有多少人在使用没有一个明确的概念，15.4 的控制中心有一个天气插件，会在控制中心打开的时候从我们服务器拉取一次天气信息，所以我经常开玩笑说，如果正式版发布的时候这个服务因为太多人访问而挂了，我都不知道该哭还是笑。玩笑归玩笑，事实即使去掉我们用户中比例最大的一部分——社区用户，我们的企业用户也是很多在用我们的产品，稍有不慎可能就会造成无法计数的损失，这个我们都应该做到心里有数。

大家只能各自为战，一个萝卜一个坑，掉进去就出不来了。15.4 不同的是，除了控制中心这个庞然大物和其他一些界面调整是桌面组的贡献外，其他的功劳还要归到文件管理器、看图、安装器和音乐等项目组的身上，有时候一个项目的可能是几个项目组协作完成的。这对比起 15.3 的时候，自然是另一番境地。

责任

“众目睽睽之下，bug 无所遁形。”说得是开源社区这种“集市”模式，让所有人都可以参与到一个像 deepin 这样庞大的项目当中，协作并产生贡献。但是，我一直觉得项目协作不仅仅需要的是参与，同样重要的是任何一个参与项目的人都要有一种责任感，最基本的责任就是不要做错误的事情，什么是错误的事情？举个例子：有一个朋友跟我讲，他做完一个产品找设计师对细节的时候，设计师跟他说他拿到的设计图上的尺寸都是随便写的，还有一个最终稿……这种明知自己的工作做得不到位还要去坑别人的做法，我认为这就是错误的事情。另一个就是互相监督，尽量友好地阻止别人做错误的事。比如你是开发，知道需求有漏洞，就不要图省

说了这么多，无非就是一句，要相信“坚持做正确的事”可以避免自己和团队走过多弯路，最终受益的还是你自己。

轮回

每次系统发布前都像是一次战争，每次系统发布又像是一场轮回。大的轮回是系统功能重新定义、设计、开发和发布，小的轮回是每个产品的周期性新功能、新设计、开发和维护，这就像是一个任何具有活力的生命一样，永无止境。deepin 就是这样的轮回中，不停的更新和优化自己，让广大的用户可以少些折腾，多些快乐。

最后希望 deepin 在以后也能继续绽放光彩！

我是特意注册一个号来支持 deepin 的

adamandeve / 2016-12-12 11:26

浏览: 322 / 回复: 11

平心而论，深度是我使用的国产系统中最棒的
比起那些骗 ZF 经费的所谓国产 OS
深度是用心在做的
深度会考虑到桌面用户的体验
让那些拿个国外的 linux 发行版，改个图标，改个主题就自称为 OS 的企业、产商真是无地自容
很遗憾很晚才接触到深度
以前用各个公司的发行版，包括 gentoo,Debian,Arch,opensuse,ubuntu 等等
感觉 deepin 是确确实实在做适合中国人自己的系统的
作为一名中国用户，希望深度越来越好
也希望深度不忘初衷，实实在在的打造出最完美的国产 LINUX

moonlimb 发表于 2016-12-12 11:42:06 | 只看该作者

deepin 确实很好，在国内口碑高。个人觉得：愿意倾听用户的 os 开发者才能开发出用户更满意、更喜欢的 os。那么它离一统江湖也就不远啦。

Guumi 发表于 2016-12-12 12:55:04 | 只看该作者

LZ 不多说了，除了现在必须的工作平台不能转入 deepin 以外，我个人电脑已经运行两年多了，而且除了娱乐，一般的办公都是在本电脑上。害死人的 google gears 啊！

usckur 发表于 2016-12-12 14:48:03 | 只看该作者

主力系统已经是 Deepin

udfk007 发表于 2016-12-12 19:16:25 | 只看该作者

欢迎深友。希望 deepin 越来越好。

duanchi 发表于 2016-12-12 19:59:36 | 只看该作者

我的电脑装了正版升级的 win10 和 deepinos。还是 deepinos 超级流畅，偶尔卡死。日常用没有问题的啦！哈哈

doudou586 发表于 2016-12-15 08:52:25 | 只看该作者

同意 LZ 意见，已完全使用 Deepin 办公 2 周，同时也影响同事N人使用。

体验大半个月，发现离不开了

luoyeqingzhou / 2017-1-2 23:01

浏览: 649 / 回复: 13

因为专业用的软件都在 windows，所以工作都是在 windows，完事后，我已经习惯切换到 deepin。从一开始就很顺利，做盘安装升级一气呵成，没遇到什么问题，平时也就是上上网、看看视频、听听歌，deepin 这些完全能够满足我。经过这些时间体验，对比了下 windows，抛开软件的兼容性，deepin 在易用性方面做的真的比 windows 好，桌面热区功能，习惯之后非常方面，比 windows 好用多了。10 年 windows 使用经历，习惯 deepin 后开始慢慢嫌弃 windows，从一开机差别就出来，windows 一进入桌面，散热风扇就会发出一阵风声，然后安静下来，deepin 却始终安静，开机自启 3 个应用，然后安静的等着你。开关机都很快，特别关机，貌似不会超过 3s。现在特别期待 15.4，希望 deepin 能够走的越来越远，做国产操作系统 NO.1。

jingle 官方管理员 发表于 2017-1-3 07:45:37 来自移动端 | 只看该作者

谢谢楼主的支持，关于应用生态这个也是最难的，只能 jixu 前进，然后让应用厂商来开发。

myfsbch 发表于 2017-1-3 12:08:39 | 只看该作者

希望深度未来成为国标。

wanzhende 发表于 2017-1-3 15:14:15 | 只看该作者

deepin 还是很漂亮的系统，我觉得单纯论美观性，我更愿意使用 deepin，不过有时候有些行业软件的确是没有办法，只能暂时切回 windows 平台，感觉 windows 太丑了。

paul1411 发表于 2017-1-4 21:13:27 | 只看该作者

12 的本子依旧，不过在 win 下启动特慢，转而 deepin，很流畅。

johnny123 发表于 2017-1-4 21:42:13 | 只看该作者

楼主这话我信，deepin 真的好用。

支持 deepin ！！！！！么么么么么

brucegl / 2016-11-13 21:26

浏览: 491 / 回复: 11

这个话从哪里说起尼？

本人刚工作没多久吧，算是宅男，爱打游戏，也想走技术男路线，喜欢折腾。

起因想不在各种游戏上浪费青春，我太沉迷了，所以竟然想过玩个玩不了游戏的系统不就可以了吗？

所以折腾各种系统 ubuntu、linux mint、centos，但体验感始终太差，各种资料不好，死机崩溃，错误，无法引导。

直到前些年发现 deepin，特别兴奋，国产的好！可惜。。。各种崩溃和错误。。。很慢用 apu a8。

前几天 ~又用 deepin 用 apu a10

各种好用！简单配置就可以了！

过程如下：

1. 安装有一点不顺畅停在了选择那个专家模式有个锁，还有一点 gurb2 不会用引导只要出了问题只能重新装。。 其他 linux 系统没有这个锁可以随意调换分区~这一点不明。

2. 安装 shadowsock 太厉害！（推荐自己买外国 vps 国内各种奸商啊。。）发现一个问题安装一系列软件后不知那个软件占用 1080 端口然后 ss 默认本地 1080，所以总是启动 ss 不成功。 ps：不知可不可以像 win 版那样分网址设置，最好别全部翻抢。

3. 百度云未成功，经过了，有些帖子教授的更改方法。希望解决下！

4. 迅雷 xware 至今可以用，很兴奋！！

5. 浏览器用的谷歌安装 Chameleon 后可以下载 youtube 视频，好像下载网页百度云可以直接启动迅雷下载，不像 win10。

6. moonplayer 记得以前有各种插件，现在没法用了，不知为何。

7. 办公软件有网页版 office 和 wps 够用了，有一点遗憾，有时不明的会假死机鼠标可以动，别的不可以动，ctrl +alt +f1 无用。

8. 至此比较顺畅搬家到 deepin。

d123 发表于 2017-1-9 17:11:00 来自移动端 | 只看该作者

顶楼主，我们都是这么过来的。

Guumi 发表于 2017-1-9 21:09:11 | 只看该作者

为同一个想法转 deepin 来的点赞，，，，真心那个 ss 不知道怎么用，度娘好几天了也没看出个所以然来

坚持这么多年，这是一个有梦想的团队

NewOxygen / 2017-1-25 14:39

浏览: 598 / 回复: 8

很多年前上大学的时候就了解深度 Linux，也使用它完成毕业设计，如今工作了近 4 年，这个团队还存活着，曾以为这样的团队在如今的社会无法生存，但今天看了各位的成绩，我很开心你们存活了下来，界面和功能真的是在为使用者考虑，这样的团队必须给个赞。

————————————一个暗搓搓用着的用户

wangyong 官方管理员 发表于 2017-1-25 15:50:40 来自移动端 | 只看该作者

从今天开始明灿灿的用户。

rekols 发表于 2017-1-25 16:16:26 | 只看该作者

bbs.deepin.org 我眼中最好的开源社区。

xiaoshitou 发表于 2017-1-25 16:37:19 | 只看该作者

小白我也用了好多年了！

He8617439 发表于 2017-2-8 19:53:05 | 只看该作者

只要慢慢来，用户会越来越多的，就怕不能坚持。

kupo 发表于 3 天前 | 只看该作者

纯好奇心使然，接触了 deepin，遂一发不可收拾！

别的不说，就这友善的安装界面就没 sei 了！

看得出来，开发团队特别用心地在拥护着 deepin 的用户。

希望系统 bug 越来越少，团队发展越来越好 ~~~

gaoyanglion 发表于 3 天前 | 只看该作者

已是办公用系统。

xiaoxie 发表于 3 天前 | 只看该作者

非常好的系统，值得深入！

金融行业 ATM 设备操作系统与培训案例

2015 年深度科技为邮储银行定制了国产 ATM 机操作系统，并成功运营首个全国产 ATM 设备，实现了取款、存款、转账、查询等完整的存取款功能，为深度科技在国内 ATM 操作系统领域奠定了坚实的基础，后续又有众多国内外知名 ATM 厂商和深度达成了合作伙伴的关系。

2016 年末深度科技与韩国著名综合性企业晓星集团下的子公司——晓星金融设备（惠州）有限公司达成了面向金融领域层面的深入合作，主要包含深度科技自主研发的 LFS 软件、服务及培训三个方面，目前培训及相关服务合作进展良好，双方后续将会开展更深入的合作伙伴关系。

晓星集团，作为韩国七大综合商社之一，在国际享有极高的声誉。韩国株式会社晓星创建于 1957 年，从事 8 大事业领域，多项世界领先，其中 ATM 事业部位居全球第四大。该公司是韩国十大企业集团之一，在全球拥有 30 余个海外公司，其中在中国北京、上海、重庆、广州、青岛设有办事处，并在北京、嘉兴、珠海、青岛设有多家企业，是一家具有国际化视野的全球性企业。

作为全球化的综合性企业，晓星集团 ATM 事业部在全球排位位列前茅，选择与深度合作，也证明了深度科技在操作系统领域雄厚的研发实力。深度科技愿意携手各界 ATM 厂商为中国银行业 ATM 领域的操作系统国产化进程做出更多的贡献！

湖北省信息安全平台洪山试点项目

项目背景：

在信息化发展如此快的今天，各国政府对信息安全的重视已经上升到国家战略高度。早在 2014 年 2 月 27 日中央网络安全和信息化领导小组第一次会议中指出：“没有网络安全就没有国家安全，没有信息化就没有现代化。”此次洪山试点项目其目的是为了达到安全可靠，自主可控。

应用背景：

项目目标是要把政务内网中的 OA、公文流转、督查系统、会议系统、邮件服务系统、应急系统等应用迁移到国产自主可控平台。项目中硬件平台使用的是国产龙芯和申威，数据库使用的是达梦，操作系统是深度，烽火集成承接整个集成项目。

项目中遇到的困难：

1. 要与各个平台做调试和优化，在适配当中有很多问题需要和厂商不断的协调、沟通并给出切实的解决方案。

2. 与烽火集成沟通整体解决方案。

3. 项目部署时间长。

解决方案：

1. 公司内部就项目进展情况不定期的进行沟通协调，只要是现场不能够马上解决的问题，现场技术会把问题一一记录，公司项目小组根据记录问题进行反复沟通并提出解决方案，对已经和可能会出现的问题进行反复测试，尽可能更高效的配合用户及各个厂家完成迁移工作。

2. 在与外部各厂商沟通协调的时候，深度科技技术人员尽可能找到问题的痛点，不论是软件还是硬件方面并给出合理的解决方案，满足客户需求。

3. 要与多方沟通推进项目实施进度。

应用效果：

使用深度操作系统后，保证了系统的稳定性和高可用性的同时，起到了对数据库和中间件很好的支撑作用，保障了业务的正常流转，很好的实现了客户前期规划效果。

如何让进程打开足够多的文件

◉ 北京 工程部/文

我们日常办公会打开多少文件呢?

“一般开十几个 Word 吧，开多了系统就卡。”

“上网冲浪，我的机器卡，浏览器最多开了一百多个；然后死机了。”

“我是网络开发的，我的 Apache 同时处理 1000 多个文件请求”

…

对于一般的应用场景 1000 个文件，已经是非常大的了。但是在一些较为特殊的场景，是远远不够的。就比如说我们最近支持的东方通中间件，在业务系统中要并发支持不低于 4000 个响应，最大打开文件数要支持到 65536 个。这对于普通的办公用户是不能想象的，但是在大型业务系统中却是司空见惯。

在春节前的最后一个工作日，某部委云平台建设项目运维人员通知我们：使用他们惯用的老办法始终无法将进程打开文件的限制数增加到 65536。他们在网上找了很多资料和方案，都说就是这么设置的；是不是深度操作系统服务器版功能有问题，或者是设置方法不一样?

我们对运维工程师提出的问题进行了详细了解，并确定他们的设置方法是生效的。同时积极配合他们，对现场采集的 lsof 数据进行分析。技术上来说：一般的场景，Linux 系统的最大文件打开数有两个限制，一个是软限制，一个是硬限制。可以通过如下命令进行查看：

```
ulimit –Sn 查看的是软限制，软件可以根据需要自行改变
ulimit –Hn 查看的是硬限制，除非具有根权限，否则不能超越这个限制
```

当然也可以通过命令对这些限制进行临时的修改，比如：

```
ulimit –Sn 65536
ulimit –Hn 65536 需要根用户权限
```

运维人员选择的方案基本上是没有问题的：在真实的业务系统中，临时的设置是不起任何作用的，而是需要对系统进行配置，来保证生产系统中特定的应用程序能够打开足够多的文件。所要修改的文件是 /etc/security/limits.conf，配置打开文件数限制如下：

```
*  soft  nofile   65535
*  hard  nofile   65535
```

重启后使用 ulimit –a 查看，发现 65535 的修改生效了。通常情况下应用就可以突破限制，达到或接近 65536 这个上限。

但是，在经过春节期间 7 天的运行，从监控情况来看现实的结果是对于部署的业务系统，并没有达到预期的效果。

我们再次与运维人员进行了深入的交流和沟通，问题是这样的：在用户处部署的系统中，部署的东方通中间件 TongWeb 运行一段时间后打开了 4000 多个文件，系统就拒绝再打开文件了。原则上我们认为是有其他的配置或应用系统引起了外部限制。但是作为深度的工程师，协助客户排查问题是我们服务的重要内容。我们逐一的查询用户打开的文件，采集 lsof 的进一步数据：

```
lsof –n lawk '{print $2}'lsort luniq –c lsort –nr
```

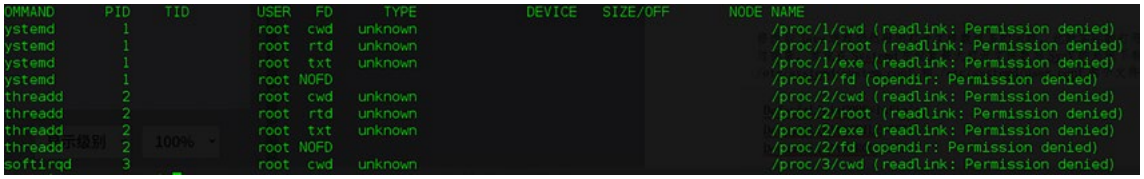


图 2

得到如下结果（图 1）：

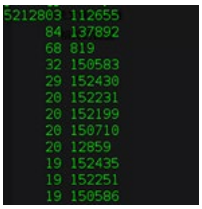


图 1

居然有 500 多“万”个文件句柄。这是什么情况? 在 lsof –n 查看（图 2）。

仔细查看，原来是是更为详细的 TID（线程）信息引起了打开文件数的重复计数，统计就不准确了。那该怎么办呢?

根据 TongWeb 程序的特性，我们发现不同的 TID 引用了相同的文件描述符；而 Linux 同一进程的线程中，文件描述符是共享的，不是单独占用文件描述符。我们找到被 TongWeb 打开的同名文件（可能是一个个 socket 链接）：

```
lsof –n lgrep 112655 lgrep "GSZJ_IDC_GSXT_WB_QTWEB_006:43464–>api.geetes t.com:http" lwc –l
```

得到一共 1212 个。也就是说，同一个文件描述符被 1212 个线程共享，并且这个共享都都被重复的计算在上面的统计中。进行计算：500 万 ÷ 1212=4125，这个就接近于限制了（因为是粗算，所以数值并不代表精确结果，只有方向指导意义）。确实是进程被限制无法打开更多的文件描述符。确定排除了双方在信息统计数据上的分歧。

问题到底出在那里呢? 经过与客户运维人员详细沟通，我们在生产系统中采集了大量进程的数据，发现了一个奇怪的现象：系统多次启动，都会以进程号 730 为分界线，进程号小的受限于 4096 个文件限制，进程号大的则限制为 65536。问题终于有眉目了，原来 ulimit 的限制是在启动的某个环节之前没有生效。而 ulimit 的配置正是在生效之后才起到了更新配置的效果。

在逻辑关系上，系统登录认证中 pam_limits.so 关联了 /etc/security/limits.conf 配置信息。这个过程之后的进程才受配置影响。在基于 systemd 的 Deepin 系统中，也只有 PAM（系统登陆认证过程调用）调起后，才生效。而系统部署的 TongWeb 中间件正是部署为系统服务，而非用户进程，并且是以 systemd 作为系统服务在用户登陆前就早早被调起了。

那么要想 TongWeb 中间件打开文件句柄支持到 65536，设置 ulimit 参数肯定是没有效果的。一定要对应的调整 systemd 的配置，而不仅仅是 ulimit 配置。需要修改 /etc/systemd/system.conf 和 /etc/systemd/user.conf 两个文件都添加如下内容

```
DefaultLimitCORE=infinity
DefaultLimitNOFILE=65535
DefaultLimitNPROC=65535
```

重启系统，然后再查看如下信息：

```
cat /proc/[PID]/limits
```

此时 TongWeb 的 java 进程的最大打开文件数为 65535，应用也不再报打开文件数过多的问题。

深度科技技术支持工程师始终把用户服务和用户价值作为产品核心价值的一部分，工作中始终保持着以用户需求和用户需要为导向的服务目标，赢得了用户运维人员和运维团队的一致认可。

深度桌面操作系统架构设计

武汉 王勇

概述

为了让各位开发者更好的了解深度操作系统，本文将以图文并茂的形式，讲解深度桌面操作系统的架构设计和基本模块的功能，帮助开发者纵观全局，更好的理解设计理念和方向，代码实现和模块依赖细节在此忽略，详细代码请各位开发者以 https://github.com/linuxdeepin 中的代码实现为准。

架构总览

一个完整的桌面系统从技术剖面看，从下到上主要分这几层：

内核驱动层	主要用于驱动硬件，除了 CPU、内存、磁盘外，最主要的要是要广泛兼容不同的网卡、显卡、声卡和外设等硬件设备。
显示服务层	从内核引导到 plymouth（我们俗称的开机动画）后，只要你见到登录界面输入密码的时候，这时候 X Server 已经起来了， X Server 简单来理解就是 Linux 系统中掌握着绘制图形界面生杀大权的“天神”，所有程序要绘制图形的时候都要发送消息到 X Server，X Server 才会给你画出来。同时 X Server 也是事件输入（键盘鼠标）输出（显示器）的抽象层，开发者可以不用考虑底层驱动和显卡驱动细节，直接就可以使用 X11/XCB 的 API 进行应用开发，只不过更多的开发者是使用 Gtk+/Qt 这些在 X11/XCB 更上层的 API 进行应用开发。
显示管理器	简单的理解就是你看到的登录界面提示你输入密码的那个位置。
资源管理器	主要由一系列的底层守护程序来监控硬件的状态，并汇报给上层的桌面环境和应用进一步操作，比如常见的就有网络、电源、磁盘、蓝牙、声音、键盘、打印等。
桌面环境	以深度桌面环境为例，主要包括桌面环境后台服务和守护进程、桌面环境对外提供图形开发工具库、二进制工具、DBus API 服务和桌面环境 UI 界面层几个部分组成的，后面我会详细讲每一个细节。
应用商店	主要提供系统的软件安装、卸载、升级等操作，保证用户可以安全方便的进行软件管理，同时提供了商店的评论和评分等功能。
应用程序	主要包括深度开发的系列应用、合作开发的国内应用、Android 应用、Windows 应用和网页应用。

显示管理器

首先看显示管理器，当 X Server 启动以后，根据系统启动服务的顺序，显示管理器就在 X Server 之后启动。深度系统使用的是由 Ubuntu 开发的 LightDM，其他主流的显示管理器还有 Gnome 的 GDM 和 KDE 的

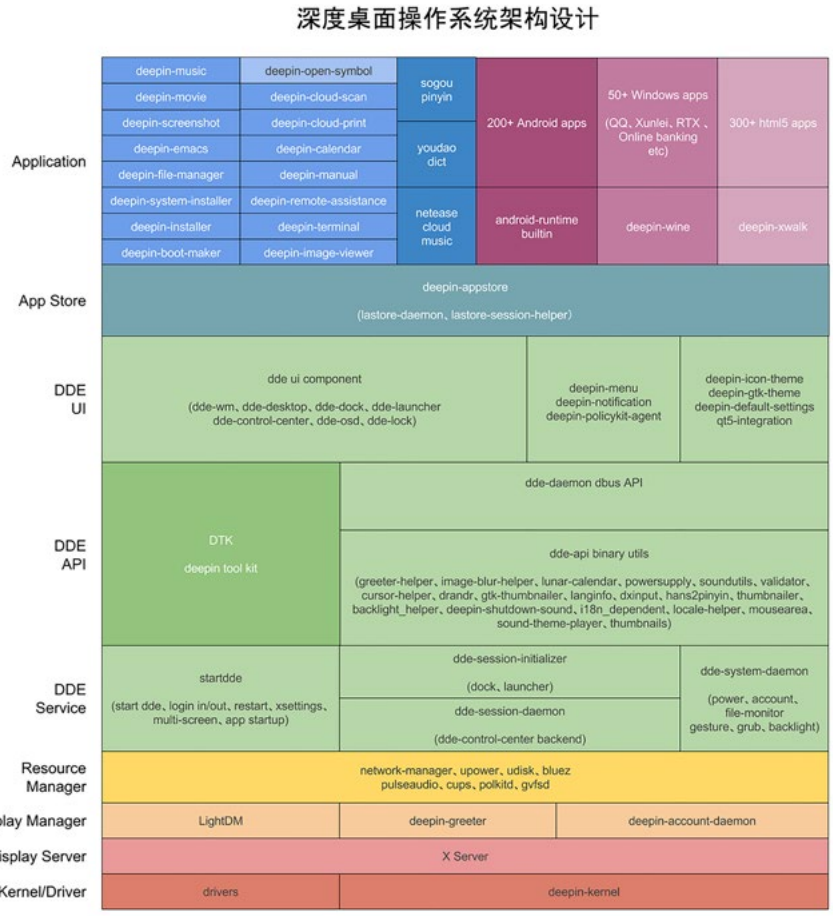


图 1　架构总览

KDM，使用 LightDM 的原因主要是 LightDM 非常的轻量，不绑定任何桌面环境，而且提供 Gtk+、Qt、Html5 等各种前端界面的定制接口，非常方便。

显示管理器主要是根据系统中已配置用户的权限对正在登录的用户提供权限认证和多用户切换功能，一旦认证通过后就从 greeter (LightDM 定制的配置接口）中执行下一步启动程序（通常是桌面环境的初始化程序），以显示桌面环境。

Deepin 开发了一个基于 Qt5 的前端界面程序，deepin-greeter 主要长这样（图 2）：

除了基本的用户认证、多用户切换、日期和关机功能以外，还会提供：

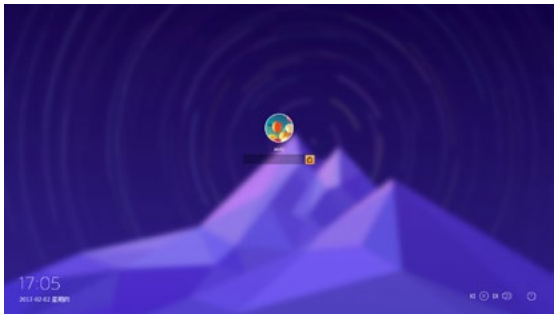


图 2　deepin-greeter

· 多媒体的控制接口（右下角），方便用户锁屏和切换用户的时候切换音乐和暂停音乐；

· 当系统使用多屏幕时，启动和解锁时，会根

据用户的鼠标位置切换锁屏主界面；

· 在用户输入密码的时候，就直接启动一些系统级的服务，比如电源、账户、亮度等守护程序，这样用户输入密码的过程很多系统服务就加载好了，相对于其他桌面环境同等服务缩短 30% 以上的登录时间。

DDE 后台服务

当显示管理器认证成功以后，就会调用 greeter 的 exec 参数，深度桌面环境就是 startdde，startdde 从名字看很容易理解，就是启动深度桌面环境的引导程序，为了让深度桌面环境可以正常启动和显示，调用 startdde 时会执行以下操作：

· 基本的桌面会话管理，比如大家熟知的注销、重启等操作；

· 按照 xsettings 主题规范设置整个系统的主题，保证桌面环境以及桌面环境的应用，不论 Gtk+ 还是 Qt 都可以正常的加载用户设置的主题。屏幕 DPI 设置也是在这个阶段初始化完成的；

· 根据 freedesktop 的各种规范，规范应用程序的启动方法，包括执行 *.desktop 文件的命令，启动提醒等；

· 多屏的管理，保障桌面环境在多屏情况下，可以在正常的主屏显示任务栏和桌面程序；

· 开机启动服务的顺序管理，比如会优先启动 dde 核心组件，才运行启动其他应用程序，防止所有开机程序在登录的一瞬间同时启动，而这时候往往很多系统服务（比如 DBus）都还没准备好，大家一团乱抢 CPU 资源不但无法快速启动，还会导致其他程序都无法启动，想象一下 3 个人同时挤一个公交车门是什么状态？这时候 startdde 就是登录后到 dde 守护进程启动之前的裁判，只有它授权的程序才能启动，没有授权的都进入暂缓状态，直到更优先的程序启动完毕。

startdde 启动以后，首先会启动 dde-session-initializer 这个进程，这个进程的主要目的是提供给任务栏和启动器提供后台服务，主要包括：

· 任务栏和启动器的有那些常驻程序；

· 所有应用程序的启动状态维护；

· 应用程序所在工作区和位置的状态维护；

· 当前系统中所有安装应用程序的图标、启动状态维护。

dde-session-initializer 就相当于任务栏和启动器的后台守护进程，提供了任务栏和启动器的核心功能，如果没有这个程序，我们日常的应用图标点击、开机启动包括程序窗口的切换都无法进行。

dde-session-initializer 本来是 dde-session-daemon 的一部分，为了加快用户登录到桌面的速度，从 dde-session-daemon 中分离出来，用以加速任务栏和启动器的显示。

dde-session-daemon 和 dde-system-daemon 就是整个桌面操作系统的后台守护程序，这两个进程维护了所有硬件的状态，包括前面说的网络、电源、磁盘、蓝牙、声音、打印、授权、共享文件、键盘鼠标等，相当于对资源管理器的各种守护进程进行了更高层次的代码封装，把面向底层硬件状态的接口转换成面向用户设计导向的接口。

根据 Linux 的最小权限划分，又把所有的后台服务分成 dde-session-daemon 和 dde-system-daemon 两个进程，dde-session-daemon 只掌握那些不需要超级权限的功能模块，比如声音、键盘鼠标、日期时区等。dde-system-daemon 掌握那些需要超级权限的功能模块，比如电源、账户、文件操作、亮度等。通过超级权限的不同进程沙箱的划分，保证执行超级权限的进程被限制在最小化的范围，避免因系统权限传导而导致的很多安全事件。

dde-session-daemon 和 dde-system-daemon

做的事情举例说明：

· 提供用户的创建、删除和管理功能；

· 管理多个屏幕的不同状态，包括位置、方向、分辨率和亮度等；

· 管理不同文件类型的默认程序和主题设置；

· 管理网络的有线、无线、VPN、DSL 等网络设置；

· 管理蓝牙、声音、日期、时间时区等设置；

· 管理电源、键盘鼠标设置；

· 管理系统的升级和 grub 设置；

· 提供多点触摸板手势的服务。

dde-sessin-initalizer、dde-session-daemon 以及 dde-system-daemon 从功能上，相当于 Gnome 的 gnome-session-daemon 所做的事情，只不过深度团队根据用户的需要以及很多优化加速设计，用 golang 重写了整个后台守护进程的代码。可以说 DDE 和 Gnome 以及 KDE 一样，都是调用底层的库 (network-manager、upower、udisk、bluez、pluseaudio、cups、polkitd、gvfsd) 对桌面环境和应用提供更为抽象和高级的服务。

DDE API

在深度桌面环境的后台守护进程基础之上，桌面环境会对外提供一个 API 层，包括图形开发工具库、二进制工具和 DBus API 接口，供桌面环境和应用程序直接调用，而不用自己重头开发，其中 DBus API 部分都通过 DBus 总线在应用调用特定的接口时动态唤醒（默认不常驻内存），任何语言编写的应用都可以轻松调用，根据上面图所示，从左到右分别进行介绍。

DTK

DTK (Deepin Tool-Kit) 是基于 Qt5 开发的一整套 UI 图形库，方便统一的编写深度桌面和深度系列应用，主要的功能有：

· 提供单实例的接口，方便直接使用，不用造轮子；

· 提供 XCB 窗口移动、缩放等一系列函数，无边框的窗口不用自己折腾几大本 X11/XCB 的书了，开发者全部都做好了；

· 提供一大票美观的自绘控件，不用自己造 Qt 控件了，拉着直接用。

感兴趣的开发者查看源代码：https://github.com/linuxdeepin/deepin-tool-kit，基于我们的 DTK 比直接基于 Qt5 开发，能够更快的开发出美轮美奂的产品，同时也欢迎社区开发者大神吐槽和提交补丁。

dde-api binary utils

主要是 dde-session-daemon 和 dde-system-daemon 在开发过程中发展出来的二进制工具，方便深度桌面环境以外的应用可以直接使用这些工具，减少核心技术的重复开发：

· greeter-helper：提供锁屏界面的语言，键盘布局，主题等内容的设置接口；

· image-blur-helper：提供壁纸模糊服务，你可以通过这个服务快速模糊一张图片，而不需要自己编写模糊算法，深度团队做的模糊算法，即使在龙芯芯片上都只需 30ms 的时间，要远远快于社区的模糊代码的性能；

· lunar-calendar：提供日历查询服务；

· powersupply：对电源接口的更高层封装，使用 `udev` 来获取电源状态以及电池信息；

· soundutils：提供了播放桌面音效的相关接口；

· validator：用户名正确验证器，不用自己编写一大堆正则表达式来做这件枯燥的事情；

· cursor-helper：提供了光标主题的设置接口；

· drandr：对 `x11 randr api` 更高级的接口封装，提供显示器的详细信息；

· dxinput：对 `x11 xi/xi2 api` 更高级的接口封装，提供输入输出设备的属性获取及设置功能；

· 后面还有很多其他高级服务，都是由 https://github.com/linuxdeepin/dde-api 提供的，欢迎各位社区开发者研究，扩展其玩法。

dde-daemon dbus API

这一部分主要是由 dde-session-daemon 和 dde-system-daemon 提供的 DBus 接口给深度控制中心前端界面使用的，外部应用程序也可以直接使用这部分 API 来快速开发，而不用自己研究和编写与系统底层软硬件打交到的代码，简单的说几个功能，感兴趣的朋友可以直接查看深度控制中心的界面代码：https://github.com/linuxdeepin/dde-control-center。

· 查询当前系统有几个屏幕，哪些屏幕是主屏，分辨率是多少?

· 查询当前系统的语言、亮度、音量等设置；

· 查询当前系统的网络链接状态：连接的是无线还是有线，有没有开启 VPN?

· 查询当前系统的日期时间、时区、键盘鼠标等外设的状态。

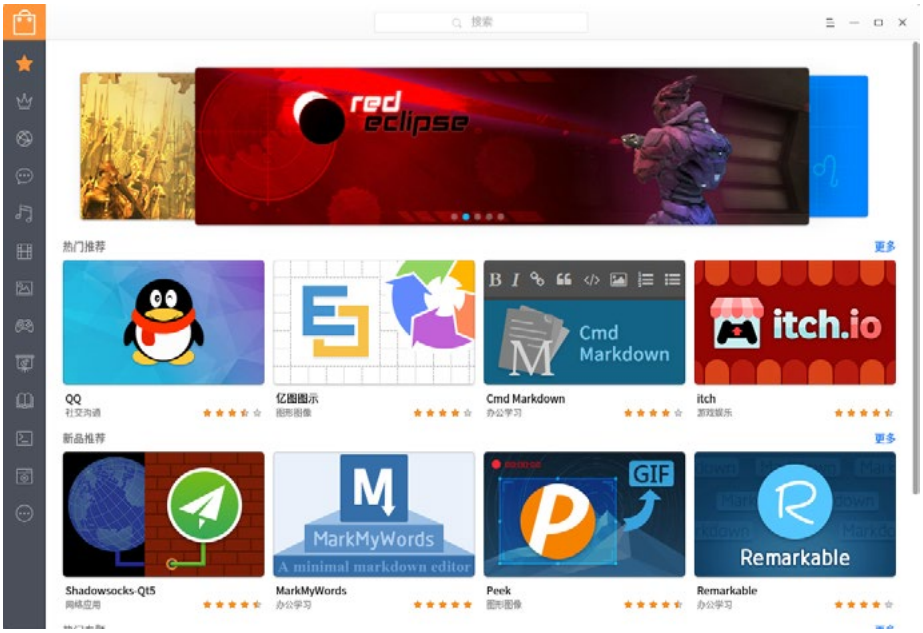
只要控制中心界面显示的所有硬件状态，都可以通过 dde-api 提供的 DBus 接口服务查询到，而这些 DBus API 后面的源代码都是深度操作系统研发人员经过非常多的时间打磨好的，不用自己痛苦的去裸写底层库 (network-manager、pluseaudio、bluez、upower、udisk 等) 代码，大大节约了应用开发者编写高级功能的时间和投入成本。

DDE UI

深度桌面环境的最后一部分就是深度桌面环境的 UI 展示层，主要有：

deepin-wm	主要写了两个窗口管理器，根据用户的硬件性能和显卡驱动情况自动使用 2D 窗口管理器还是 3D 窗口管理器。其中 deepin—metacity 是基于 metacity 的基础之上开发的 2D 窗口管理器，适用于低配置电脑或者显卡驱动没有正常驱动的电脑，对系统消耗的资源比较小但是动画效果比较生硬，deepin—wm 是基于 mutter 编写的 3D 窗口管理器，也是官方的默认窗口窗口管理器，主要用于显卡驱动正常和高配置的电脑，同时也提供更多的动画细节让整个桌面环境动效更优雅
dde-desktop	提供桌面基本的文件网格显示和基本的文件操作，dde—desktop 通过 libdde—file—manager 这个库来保证所有的文件显示和文件操作效果和深度文件管理器 (dde—file—manager) 保持一致
dde-dock	主要提供图形化的任务栏图标管理和状态维护、在屏幕中位置以及两种风格的切换
dde-launcher	主要提供系统已经安装应用的展示、搜索和启动等功能
dde-control-center	提供对所有硬件资源控制的前端界面，包括账户、主题、网络、蓝牙、声音、外设、默认设置程序、日期时间、电源以及系统升级等模块

dde-osd	当按下音量、亮度等多媒体按键时在屏幕中间显示快速提示
deepin-menu	统一所有软件右键菜单的 UI 细节
deepin-notificaiton	通过右上角提供系统的通知服务，根据不同应用显示不同的快速跳转按钮
deepin-policykit-agent	主要按照深度的 UI 设计规范做个了密码认证对话框，保证所有程序，不论是 Gtk+ 还是 Qt 写的，在密码验证的时候都弹出 UI 细节一模一样的对话框
deepin-icon-theme	世界上风格最统一，数量最多的图标，包括应用商店上千款应用的图标，具体请查看 https://github.com/linuxdeepin/deepin-icon-theme/tree/master/deepin
deepin-gtk-theme	主要是针对深度自己的设计规范来重新制作的整套窗口主题（Gtk+2、Gtk+3、Qt4、Qt5）
deepin-default-settings	深度桌面环境默认的设置，比如默认的壁纸、图标主题等默认设置
qt5-integration	给 Gtk+ 和 Qt 源码编写了补丁，保证所有程序弹出的文件打开对话框都是完全一致的体验，不论是 Gtk+ 还是 Qt 编写的程序，再也不用为不同图形库开发的程序弹出不同风格的文件打开对话框而烦恼



深度应用商店

深度商店是 Linux 下第一款有产品质量和交互体验优秀的应用商店。

除了支持深度系列应用，还通过 deepin-wine 支持 50+ 多款 Windows 应用，通过内置 Android runtime 支持 200+ 多款 Android 应用(比如愤怒小鸟，各种视频客户端)，通过和 Intel 合作开发 deepin-xwalk 直接支持 html5 应用 (比如 Gliffy 等) 还可以自己记住窗口大小 (而不是打开一个贼大的浏览器)

深度应用商店从技术架构上，主要分为 lastore-daemon、lastore-session-helper、deepin-appstore 三个部分：

· lastore-daemon：应用商店的核心部分，主要负责所有软件的安装程序的下载、哈希安全校验、依赖分析、本地缓存管理、软件安装、软件卸载和软件升级等工作，这个守护进程会和深度的软件仓库智能的通信，在后台保障整个操作系统应用的更新和安全守护。lastore-daemon 同时会在 apt/dpkg 程序中加入钩子，不论用户是从应用商店还是终端中安装的程序都会被商店守护进程管理和保护，避免一些高级用户从终端安装以后把系统的依赖弄坏。

· lastore-session-helper：上面说的 lastore-daemon 默认就会有超级权限以进行软件包的管理，但是一些普通的用户会话级的操作，比如安装成功以后通过右上角通知提醒用户，本地化管理这些操作都是不需要超级权限的，为了最小化超级权限代码的执行范围，最大程度保证用户安装软件的安全，开发者就从 lastore-daemon 中剥离了这部分代码放到 lastore-session-helper 中以普通权限来执行。

· deepin-appstore：这个就是大家上面看到客户端部分的代码，简单来说就是一个 CEF 框架基础上构建的应用程序壳，处理客户端本地的用户交互然后嵌入一个网页，商店的服务器一旦更新了新的软件后，就会通知 deepin-appstore 进行页面刷新。还包括评论、评分等操作的界面接口。从技术的角度来形容，deepin-appstore 就是一个具有本地客户端操作和样子的简易浏览器。

我与 VirtualBox 的那些事儿

北京 工程部

今天给大家讲讲我和 vbox 的那些事儿，同学不要想歪，vbox 不是人，vbox 是一款软件，一个你可以在它之上安装很多系统的软件，听到这里大家有木有很感兴趣呀，继续说，vbox 做为开源阵营中一款重要的虚拟机，受到了许多源粉的喜爱，最主要它是轻量级的虚拟机，即使机器配置不太高的同学也能使，占用资源少，正版也免费。如果只是简单的系统操作，建议使用 vbox，而且 vbox 在 deepin 系统中运行还是非常稳定的，但是 vbox 在实现某些功能时会出现一些小问题，让我们踩了很多坑，今天就给大家说说我们怎么填满那些坑的。

介绍完了 vbox 是什么，那就先说说它的安装吧，在咱们专业版版操作系统上是 vbox5.0.12，在社区版操作系统上是 vbox5.1.10，这里先说 5.0.12，其实 vbox 的安装非常非常的简单，我相信这对所有的小伙伴来说都不是问题，在深度商店的搜索框中输入 VirtualBox 或直接虚拟机，回车就可以看到 vbox，单机打开它选择安装就好，安装完成后在启动器中就可以看到 vbox，单击打开它，就可以开始使用这个软件。

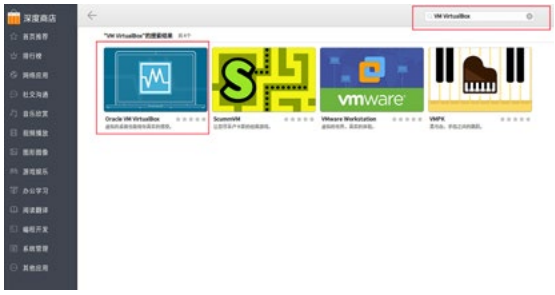


图 1：安装即可

由于有些小伙伴因为工作原因需要使用 Windows 系统，所以我们需要在该软件中新建一个 Win7 的虚拟机，新建虚拟机过程我这就不详细说了，安装软件和新建虚拟机这都没什么问题的，vbox 如果你要实现它某个功能必须得安装它的扩展包，我们最需要实现的功能就两个，一个是文件共享（虚拟机中的系统与本机互传文件数据），另一个就是虚拟机里的 Win7 可以识别到 u 盘。其实要实现文件共享这个功能也是非常简单的，安装步骤简单给大家说说吧：

1. 在该软件上创建一个 Win7 系统并启动它；

2. 在虚拟机上方功能菜单中可看到“设备按钮”——> 安装增强功能；

3. 单机“安装增强功能”开始下载。

4. 在下载完成后在 Win7 的文件管理器中选择 vbox 驱动开始安装。

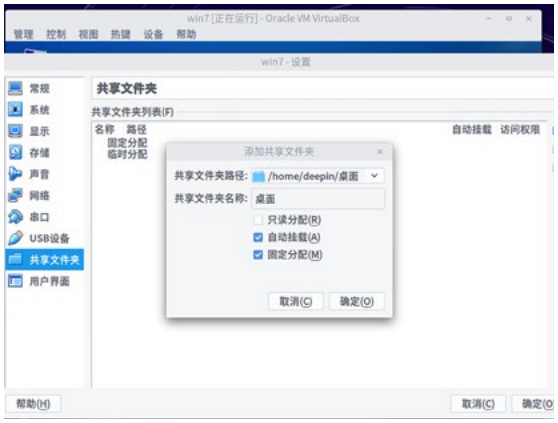


图 2：在“共享文件路径路径”中可自由选择你要共享本机的那个位置

5. 在“设备”——>“共享文件夹”中即可共享本机磁盘、文件等等。

到这里大家是不是在想你个骗子，不是告诉我们你踩了很多坑，这不挺顺利的嘛，别着急，这不还需要实现另一个功能嘛——虚拟机识别 usb，说到这个问题，等会儿让我哭五分钟，1 ~ ~ ~ ~好，我回来了，各位同学你们辛苦了，听我费了半天话，开始给大家讲讲我们踩的坑，这样大家以后就不会再掉入坑，因为着急解决这个功能，所以张老板也一起参与测试这个功能。

也是同样的想要实现这个功能就需要安装一个扩展包，这个扩展包名叫：“Oracle_VM_VirtualBox_Extension_Pack–” 版本号“.vbox–extpack”扩展包可以到 vbox 官网下载，扩展包的版本号一定要与 vbox 对应上，安装方式：

1. 将安装包拖到虚拟机中的 Win7 系统桌面；

2. 双击该扩展包，选择安装；

3. 安装完成后可在 vbox 首页有一个管理——>全局设置——> 扩展中可看到已安装的扩展包；

4. 选中该扩展包，选择确定按钮。

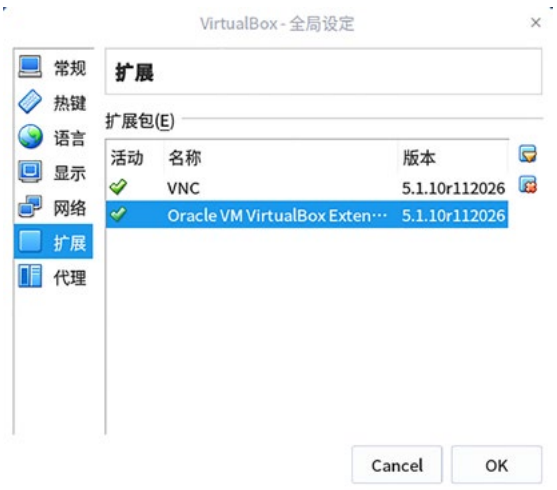


图 3：在“扩展”中可看到已安装的扩展包

安装完成后，在 vbox 首页的“设置”中找到 usb 设备，勾选 usb3.0 或 usb2.0 识别器后，开启虚拟机时，报错：“无法为虚拟服务 Win7 开启一个任务”，这时又重新创建一个 linux 系统的虚拟机，添加完镜像，开始安装系统时，仍然报错“无法为虚拟服务 linux 开启一个任务”，上网查了查资料，可能是受版本的影响。

正好旁边有台测试笔记本是社区版操作系统，下载了 vbox5.1.10，安装 usb 的扩展包“Extension_Pack”并开启识别 usb3.0 的功能，开始创建一个 Win7 虚拟机，发现在安装系统时没在报“无法为虚拟服务 Win7 开启一个任务”这个错，在系统成功安装完成后，心情很激动，单纯的我以为这样就可以实现这个功能了，赶紧插上 u 盘试试，咦？怎么识别不了，什么鬼，冷静一下，确定了版本号是相同的，安装也没问题，于是决定上网查查资料，在这里我想要先吐槽一下，vbox 在 2015 年底就开始在论坛讨论识别 usb3.0 的这个问题了，但是在论坛中 vbox 给予免费版的回复比较少，主要在解决商业版的问题，所以查找的处理结果都不是那么完整，接着说：在某论坛还算是找到了一点儿有用的资料，需要将计算机当前登陆的用户加入 vbox 组里，还需要新建一个 usbfs 的组并将当前登陆用户加到该组里，方法如下：

图 4：sudo addgroup usbfs 创建 usbfs 组

sudo vim /etc/group 使用”vim“编辑用户组的配置文件

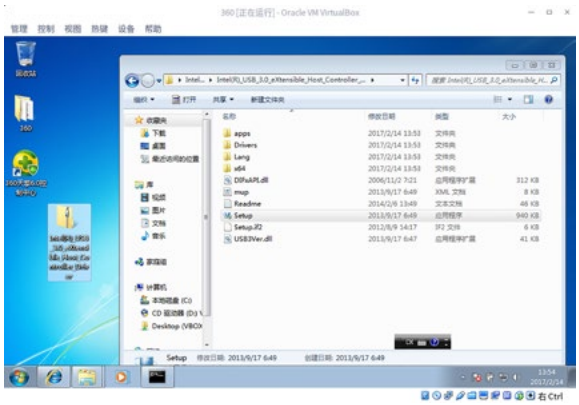


图 6：解包完成后如图，双击”Setup“安装即可

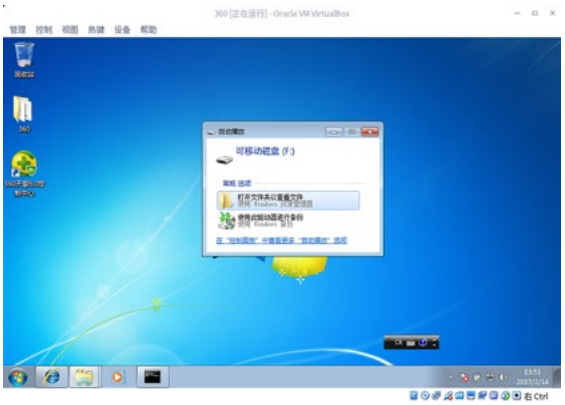


图 7：完成以上操作，即可识别到 u 盘

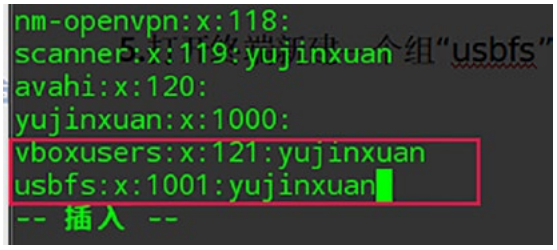


图 5：将当前用户”yujinxuan“加入”vboxuser“和”usbfs“组里

做完这些操作，又怀着激动的心情重启计算机、打开 vbox 软件开启 Win7 系统、插上 u 盘，后面发现还是不能识别 u 盘，又上网查了资料，做了操作还是不能识别 u 盘，后面想这是不是 vbox 版本的有问题，卸载了这个版本，重新下载别的版本，进行测试，又开始报各种错，测到 vbox5.12 的时候，张老板发现我们忽略了一个大问题，Win7 不支持 usb3.0，需要在 win7 中安装 usb3.0 的驱动，安装完成后又开始进行测试，发现只要是在启动虚拟机前开启 usb3.0 功能，就能识别到 3.0 的 u 盘，再后面测试过程中发现 vbox5.1 以上的版本安装扩展包、usb3.0 的驱动、创建 usbfs 组并把系统当前用户加入该组和 vbox 的组，基本就能实现这个功能。

Usb 驱动安装方法：

下载 usb 驱动包并拖到 Windows 系统中解包安装。

Intel(R)_USB_3.0_eXtensible_Host_Controller_Driver.zip

最后还有一个小问题告诉大家，因为 vbox 自带的 dkms 在不同的内核版本上可能编译不通，所有不同的 vbox 在不同的内核版本上可能不一定能正常搭配使用，暂时 vbox 自带的 dkms 在咱们系统上暂时不能升级，但是对识别 usb 这个功能不会有太大的影响。

内核模块开发入门

◉ 北京 研发部

一、hello world

Linux 内核模块是运行在 Linux 内核空间的模块，它可以执行高权限操作，包括访问甚至修改内核里面的数据，也可以随着内核一起运行。但是内核模块的开发对我们来说还是一件陌生的事，这几天刚好项目需要，正好学习了一下，把心得写下来与大家分享。

学习任何开发的时候，第一件事就是要写一个 hello world 的程序，这几乎是约定俗成的事了，废话少说，放码过来。我们先把代码列举出来，如表 1、表 2 所示。

本内核模块编译和测试的步骤如下：

1. 安装 gcc、make 和 linux 内核头文件，即 sudo apt-get install gcc make linux-headers-$(uname -r)

2. 创建一个目录，并将源代码文件与 Makefile 存放在此目录下

3. 进入此目录编译代码，即运行 make

4. 按 Ctrl-Alt-F2 切换到 tty2 下，并登录系统，进入此目录

5. 运行 sudo insmod hello.ko 挂载此内核模块，你将看到 hello world，在前面还会有被中括号包围的时间戳

6. 运行 lsmod | grep --color hello 可以看到 hello 内核模块已经被载入

7. 运行 sudo rmmod hello 卸载此内核模块，你将看到 bye~ world，在前面还会有被中括号包围的时间戳

表 1 hello.c 源代码

```
#include <linux/kernel.h>
#include <linux/module.h>

int init_module()
{
    printk(KERN_ALERT "hello, world\n");
    return 0;
}

void cleanup_module()
{
    printk(KERN_ALERT "bye~ world\n");
}

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Raphael");
MODULE_DESCRIPTION("Hello World Kernel Module");
```

表 2 hello world 的 Makefile

```
obj-m += hello.o

kdir := /lib/modules/$(shell uname -r)/build

all:
    make -C ${kdir} M=$(PWD) modules

clean:
    make -C ${kdir} M=$(PWD) clean
```

恭喜你，你的内核模块的第一个程序已经顺利运行了（此处应有掌声 ;）！

下面我们来一一解释下代码里各行的意义，免得大家一头雾水，知其然不知其所以然。

首先是 hello.c 的源代码。在源代码里，首先需要注意的就是我们没有 include 任何普通 C 程序的头文件，例如 stdio.h、string.h 等。这是因为在内核态编程，我们不能使用 glibc 的任何函数，因为显然 glibc 的很多函数是依赖于 Linux 的系统调用的，而

系统调用又是给用户态的程序提供的内核的接口，在内核里调用用户态的代码显然不靠谱。实际上，在内核模块开发过程中，我们 include 的基本上都是 linux/ 开头的各头文件。

其次我们需要注意的是此代码没有 main 函数，只有 init_module 与 cleanup_module 函数，从字面上，也从运行效果上来看，前者显然是在本内核模块被载入 (即 root 运行 insmod hello.ko) 时被调用的，后者则应该是本内核模块被卸载 (即 root 运行 rmmod hello) 时被调用的。在此处，你实际上也可以定义自己的初始化与卸载函数名，只要满足相应的函数签名要求，然后通过 module_init 与 module_exit 调用设定初始化与卸载函数即可。如表 3 所示：

表 3 自定义初始化与卸载函数

```
1 int init_hello(void)
2 {
3     printk(KERN_ALERT "hello, world\n");
4     return 0;
5 }
6
7 void exit_hello(void)
8 {
9     printk(KERN_ALERT "bye~ world\n");
10 }
11
12 module_init(init_hello);
13 module_exit(exit_hello);
```

此外，大家也应该注意到了，我们使用的打印函数不是 glibc 提供的 printf 等，而是内核提供的 printk，其中前面的 KERN_ALERT 表示消息级别，是内核定义的一个常量字符串，表示日志级别的，类似的常量在 linux/kern_level.h 里都有定义。

之所以使用 KERN_ALERT 是因为它的日志级别很高，而且在 /proc/sys/kernel/printk 里缺省的 console 打印的日志级别是 (在内核代码的 kernel/printk/printk.c 里定义了 CONSOLE_LOGLEVEL_DEFAULT 等值) 是低于 KERN_ALERT 的 (注意，数值越小级别越高)，因此可以在控制台下打印出 KERN_ALERT 级别的日志来。

在这里要注意下，在 X 的模拟终端下是无法打印出来的，所以需要通过 Ctrl–Alt–F2 切换到控制台下。如果你使用的是其它级别的日志，只要高于 / proc 里设置的终端日志级别，都是可以打印在控制台下的。同时，通过 dmesg 命令或者 kern.log 文件等也可以看到这些日志，其中，dmesg 还会根据日志级别标上不同的颜色，非常贴心呢。

最后三行 MODULE_ 开头的宏分别定义了本内核模块的许可证、作者姓名与描述。这些信息都可以在模块加载后，通过运行 modinfo hello.ko 命令看到。另外，需要注意的是由于本内核模块放置在了用户目录，而不是 /lib/modules 目录下，因此直接 modinfo hello 是会失败的。此外，从 modinfo 输出的 vermagic 信息中就可以看到内核模块编译的内核环境，如果将编译好的内核模块放在一个不同的内核下安装，则会产生类似 Invalid module format (–1): Exec format error 这样的错误信息，此时解决问题的唯一方法就是需要得到驱动 / 内核模块的源代码，重新进行编译才行。这也就是为什么很多软件虽然提供 deb 包，但是却没有 dkms 的原因，因为版本太多了，还不如给用户源代码自己编译呢。

另外还有一点值得注意的是如果你没有设置许可证，或者设置的许可证不是 GPL 和其它兼容的开源许可证，则在载入此模块的时候会在日志里显示一个警告，即内核已经被污染 (kernel tainted) 了。

下面我们来聊下 Makefile。

如果你尝试过就知道，使用 gcc hello.c 是无法编译出内核模块的，因为无法找到对应的内核头文件，其实即使找到了头文件也会无法链接。因此内核模块的编译总是需要写一个 Makefile，而不是用 gcc 直接编译的。

Makefile 的第一行引用了一个 obj–m 变量，这个就是你需要编译的 .o 文件。第三行定义了一个内核头文件目录，即 /lib/modules/$(shell uname –r)/

build，此目录实际上是一个软链接，指向的是你内核头文件的所在目录，因此，需要首先安装内核头文件。而下面的 all 与 clean 这两个 target 实际上都是调用了 make 命令，首先 make 会切换到 (–C)kdir 目录，使用那个目录下的 Makefile 编译，然后再切换回来 (M=$(PWD)) 继续本地的编译，而 modules 和 clean 则是内核头文件目录的 Makefile 定义的两个 target 了，分别表示编译内核模块与清除编译中间与结果文件。

至此，内核模块 hello world 的源码就剖析完了，下面我们开始为内核模块添加一些基本功能。

二、传递参数

在用户态的程序里，我们往往可以通过命令行向程序传递参数，并通过 argc 与 argv 访问命令行参数，那么在内核模块里，我们又如何获取用户传递的参数呢？表 4 的代码演示了如何通过 insmod 传递参数给内核模块。

表 4 给内核模块传递命令行参数

```
1 #include <linux/kernel.h>
2 #include <linux/module.h>
3
4 char* name = "world";
5 module_param(name, charp, 0644);
6 MODULE_PARM_DESC(name, "Name of the user");
7
8 int init_module()
9 {
10    printk(KERN_ALERT "hello, %s!\n", name);
11    return 0;
12 }
13
14 void cleanup_module()
15 {
16    printk(KERN_ALERT "bye~ %s!\n", name);
17 }
18
19 MODULE_LICENSE("GPL");
20 MODULE_AUTHOR("Raphael");
21 MODULE_DESCRIPTION("Hello World Kernel Module");
```

在这里，要注意代码中关于 name 变量的声明和使用。特别是第 5 行与第 6 行代码。在这两行代码里，本内核模块将内部的变量声明为可修改的参数，并设定了参数的权限，在例子里此参数可以为 root 读写，并可以为其它任何用户所读取。这样这个参数就可以在用户态被设置了，首先是可以在载入模块时设置，就像用户态的命令行参数一样，其次可以通过 sysfs 设置，这样在运行时也可以设置了。

如下：

1. 通 过 sudo insmod hello.ko name=deepin 命令载入内核模块，并将 name 参数的值设置为 deepin，此时可以看到控制台下打印的将是 hello deepin!

2. 运行 sudo modinfo hello.ko 可以发现多了一行 parm: name 的参数描述

3. 运行 cat /sys/module/hello/parameter/name 命令确认 hello 内核模块的 name 参数为 deepin

4. 运 行 echo –n “China” > /sys/module/hello/parameter/name 命令将 hello 模块的 name 参数设置为 China，注意，使用 –n 参数是为了避免出现换行，而且由于参数的权限是 644，因此需要切换到 root 执行此命令

5. 运行 sudo rmmod hello 卸载模块，并注意显示的信息将是 bye~ China!

除了 charp 表示 char 的指针外，常见的很多 C 语言的数据类型都支持，例如 int、bool、long 等。此外，还可以通过 module_param_array 定义数组传参。这些宏定义和参数类型定义都在 linux/moduleparam.h 头文件中可以找到。此外提一句，如果没有第 6 行，参数就无法通过 modinfo 看到了，但是仍然可以通过 insmod 的命令行初始化，并使用 sysfs 设置与读取的。

当然，有时候，我们还需要对参数设置进行一些额外的处理。例如我们有一个变量表示交通灯信号，有绿黄红三色。在程序内部，我们一般是通过 enum 类型或者 int 类型表示的，但是对外，我们希

望能让用户看到 green、yellow 与 red 字样，也最好能让用户通过这三个字符串常量来设置参数，而不是通过 0、1、2 这样不好记忆的数字来设置。这时候，module_param 就不好用了，我们可以用 module_param_cb 来设置参数 (实际上 module_param 就是对 module_param_cb 的一个简化封装)。如表 5 的代码所示。

表 5 通过 module_param_cb 设置参数

```
1 #include <linux/kernel.h>
2 #include <linux/module.h>
3
4 char* lights[] = {"green", "yellow", "red", 0};
5 int light = 0;  // 0: green, 1: yellow, 2: red
6
7 int set_light(const char* val, const struct kernel_param* kp)
8 {
9     int user_light = -1;
10    for (int i = 0; lights[i]; i++) {
11        if (strcmp(lights[i], val) == 0) {
12            user_light = i;
13            break;
14        }
15    }
16
17    if (user_light == -1 || user_light == light)
18        return 0;
19
20    char tmp[4];
21    snprintf(tmp, sizeof tmp, "%d", user_light);
22    printk(KERN_ALERT "traffic-light: %s\n", val);
23    return param_set_int(tmp, kp);
24 }
25
26 int get_light(char* buf, const struct kernel_param* kp)
27 {
28    buf[0] = 0;
29    for (int i = 0; lights[i]; i++) {
30        if (i > 0) strcat(buf, " ");
31
32        if (i == light)
33            strcat(buf, "[");
34
35        strcat(buf, lights[i]);
36
37        if (i == light)
38            strcat(buf, "]");
39    }
40    return strlen(buf)+1;
41 }
42
43 struct kernel_param_ops light_ops = {
44    .set = set_light,
45    .get = get_light,
46 };
47
48 module_param_cb(traffic_light, &light_ops, &light, 0644);
49 MODULE_PARM_DESC(traffic_light, "Traffic light: green yellow red");
50
51 char* name = "world";
52 module_param(name, charp, 0644);
53 MODULE_PARM_DESC(name, "Name of the user");
54
55 int init_module()
56 {
57    printk(KERN_ALERT "hello, %s!\n", name);
58    return 0;
59 }
60
61 void cleanup_module()
62 {
63    printk(KERN_ALERT "bye~ %s!\n", name);
64 }
65
66 MODULE_LICENSE("GPL");
67 MODULE_AUTHOR("Raphael");
68 MODULE_DESCRIPTION("Hello World Kernel Module");
```

在上面的代码中可以看到，我们增加 4~49 行的代码，增加了 light 变量，而这个 light 变量对外的参数名则是 traffic_light，设置此变量是通过 set_light 函数来设置的，读取此变量是通过 get_light 来设置的。在这两个函数里的 kernel_param 类型参数对应的就是我们模块内部的参数 / 变量，而 set_light 的第一个参数则是外部调用者传进来的字符串，get_light 的第一个参数则是我们需要传给外部调用者的字符串。

在上面的代码里，我们使用了 C99 的语法来声明变量，即没有在函数开始就声明了所有变量，而是即用即声明，这样我们还需要修改下 Makefile，增加相应的编译选项，如表 6 所示。

表 6 module_param_cb 演示的 Makefile

```
1 obj-m += hello.o
2 ccflags-y := -std=gnu99 -Wno-declaration-after-statement
3
4 kdir := /lib/modules/$(shell uname -r)/build
5
6 all:
7    make -C ${kdir} M=$(PWD) modules
```

```
8
9 clean:
10    make –C ${kdir} M=$(PWD) clean
11
```

注意以上 Makefile 内容中仅增加了 ccflags-y 变量的声明。

结合表 5 的源码与表 6 的 Makefile，我们可以测试如下：

1. 通过 sudo insmod hello.ko name=deepin traffic_light=red 命令载入内核模块，并将 name 参数的值设置为 deepin，将 traffic_light 参数的值设置为 red，此时可以看到控制台下打印的将是 traffic-light: red 与 hello deepin!

2. 运行 sudo modinfo hello.ko 可以发现多了一行 parm: traffic_light 的参数描述；

3. 运行 cat /sys/module/hello/parameter/traffic_light 命令确认 hello 内核模块的 traffic_light 参数为 green yellow [red]；

4. 由于参数的权限是 644，因此需要切换到 root 来运行 echo –n “yellow” > /sys/module/hello/parameter/traffic_light 命令将 hello 模块的 traffic_light 参数设置为 yellow，注意，使用 –n 参数是为了避免出现换行。控制台下将打印出 traffic-light: yellow；

5. 运行 sudo rmmod hello 卸载模块。

顺便说一下，源码中使用 module_param 设定 name 参数的第 52 行代码等效于表 7 里使用 module_param_cb 实现的代码。

表 7 module_param 的等效表示

```
1 struct kernel_param_ops name_ops = {
2    .get = param_get_charp,
3    .set = param_set_charp,
4 };
5 module_param_cb(name, &name_ops, &name, 0644);
```

其中 param_get_charp 与 param_set_charp 是内核提供的辅助函数，用来读取和设置 char* 类型的变量 / 参数的。

三、使用 procfs 通讯

以上传递参数的方法已经相当不错了，你可以在载入模块的时候初始化参数，可以通过 sysfs 传递参数，也可以在设置参数的时候进行处理与响应，还可以通过命令行来读取与设置参数。

但是对于大规模的数据传输来说，上述方法不仅无法传递二进制数据 (任何参数的传递都必须使用字符串)，而且传输量较小。此外，从上面的代码也可以看到所有的数据访问都已经被封装了一层，即上述的内核模块代码在读取与写入时都没有访问用户态的内存地址空间，都是对经过转换后得到的内核态内存地址空间操作的。这虽然使得开发简单了，但是同时效率也下降了。

一般来说，我们更偏向于用 procfs 与 udev/devfs 而不是 sysfs 进行数据 / 参数的传递。其中，通过设备节点，即 udev/devfs 来传输数据更偏向于设备驱动程序，感兴趣的同学可以读读参考里面的资料来学习如何创建一个虚拟的设备，而 procfs 更常见于内核模块的数据传递，这两种参数传递方式都可以用来传递大量数据 (包括二进制数据)。

下面，我们将主要描述如何使用 procfs 节点进行内核模块与用户态程序的通讯。

要通过 procfs 通讯，首先我们必须要创建一个 /proc 下的节点，在当前的 Linux 内核里，创建 procfs 节点的函数为 proc_create，在 linux/proc_fs.h 头文件里声明。我们还是先看演示代码好了，参见表 8。

表 8 创建 procfs 节点源码

```
1 #include <linux/kernel.h>
2 #include <linux/module.h>
3 #include <linux/proc_fs.h>
4
5 #define PROCFS_NAME "dgst"
6
7 static struct proc_dir_entry* procfs_entry = 0;
8
```

```
 9 static struct file_operations procfs_ops = {
10     .owner = THIS_MODULE,
11 };
12
13 static int init_dgst(void)
14 {
15     procfs_entry = proc_create(PROCFS_NAME, 0644, 0,
&procfs_ops);
16     return 0;
17 }
18
19 static void exit_dgst(void)
20 {
21     if (procfs_entry != 0) {
22         remove_proc_entry(PROCFS_NAME, 0);
23         procfs_entry = 0;
24     }
25 }
26
27 module_init(init_dgst);
28 module_exit(exit_dgst);
29
30 MODULE_LICENSE("GPL");
31 MODULE_AUTHOR("Raphael");
32 MODULE_DESCRIPTION("ProcFS Demo Kernel
Module");
```

在这个例子里，我们使用了 module_init 等函数设定初始化与清除函数，这也是内核模块程序里更常用的方法。

在这里的源码里，我们可以看到在载入模块的时候模块创建了一个名为 dgst 的 procfs 节点，但是这个节点没有提供任何访问接口 (因为其 file_operations 类型的参数里没有设定任何访问函数)，因此实际上这个节点无法提供任何数据，外界程序也无法通过这个节点写入任何数据。在模块载入后，你可以看到 /proc 目录下有了一个 dgst 节点，但是你无论是 cat 读取或者 echo 写入 /proc/dgst 都会产生 IO 错误。还需要注意的是模块在卸载的时候调用了 remove_proc_entry 函数删除了此节点。

想要在用户态通过 procfs 读取内核模块的数据或者向内核模块写入数据，我们一般有两种方法：

· 通过文件的读写操作，即通过 open/read/write/close 等系统接口来打开 procfs 对应的文件来读写数据

· 通过文件的 ioctl 操作，即通过 open/ioctl/close 来打开 procfs 对应的文件，然后向其通过 ioctl 系统调用接口发送控制命令。

在表 8 的代码里，我们已经看到了要提供某个 procfs 节点的数据访问方式，可以通过 file_operations 结构体来设置，内核模块可以通过这个结构体提供 procfs 节点的打开、读取、写入、关闭、ioctl 等的接口函数，以供用户态程序调用。

但是添加读写功能之前，我们需要首先对用户态的文件读写方式有所了解。从用户态读取文件，典型的代码如表 9 所示。

表 9 用户态读取文件的典型代码

```
1 if ((fd = open(filename, O_RDONLY)) >= 0) {
2     char buf[4096];
3     while ((bytes = read(fd, buf, sizeof buf)) > 0) {
4         // do buf processing
5     }
6     close(fd);
7 }
```

根据 man 手册说明，程序调用 read 成功后，成功读取到的字节数会被返回 (0 表示到达了文件尾)，而且文件的偏移量 (file position) 也会被增加这么多字节数，出错的时候应该返回 –1，而且相应的 errno(如 EAGAIN 等) 也会被设置，以便调用者获取具体的程序错误。而 write 也有类似的返回描述。

也就是说，我们实现的 procfs 节点的 read 与 write 接口需要处理的参数应该有一个文件、一个用户提供的用来读 / 写的缓存区、一个缓存区的大小、一个文件偏移量以及一个返回值，而且返回值的处理应该包括 errno 的处理。

在下面，我们就根据上述 read/write 的要求为内核模块添加 procfs 的读写功能，参见表 10。我们先介绍一下模块的功能，此模块将对用户输入的数据进行操作，得到一个固定长度的摘要 (digest，缩写为 dgst)，类似 MD5 与 SHA 系列的摘要算法，不过当然为了简单起见，这个摘要算法超级简单，仅仅是字节异或而已。

表 10 实现了 write/read 接口的 dgst 模块

```
 1 #include <linux/kernel.h>
 2 #include <linux/module.h>
 3 #include <linux/proc_fs.h>
 4 #include <linux/uaccess.h>
 5 #include <linux/slab.h>
 6
 7 #define DGST_SIZE   8
 8 #define PROCFS_NAME "dgst"
 9
10 static struct proc_dir_entry* procfs_entry = 0;
11 static unsigned char seed[DGST_SIZE] = {0x6c, 0xb3, 0x47, 0x93, 0x21, 0x78, 0xdb, 0x2f};
12 static unsigned int dgst_offset = 0;
13 static unsigned char dgst[DGST_SIZE];
14
15 static ssize_t read_dgst(struct file* sf, char __user* buf, size_t size, loff_t* offset)
16 {
17     if (*offset != 0)
18         return 0;
19
20     if (size < DGST_SIZE*2 + 2)
21         return -EINVAL;
22
23     char tmp[32];
24      snprintf(tmp, sizeof tmp, "%x%x%x%x%x%x%x%x\n", dgst[0], dgst[1], dgst[2], dgst[3], dgst[4], dgst[5], dgst[6], dgst[7]);
25     unsigned int len = strlen(tmp)+1;
26     if (copy_to_user(buf, tmp, len) == 0) {
27         *offset = len;
28         return len;
29     }
30     return -EFAULT;
31 }
32
33 static ssize_t write_data(struct file* sf, const char __user* buf, size_t size, loff_t* offset)
34 {
35     if (size == 0) return 0;
36
37     char* tmp = kmalloc(size, GFP_KERNEL);
38     if (tmp == 0)
39         return -ENOMEM;
40
41     if (copy_from_user(tmp, buf, size) != 0) {
42         kfree(tmp);
43         return -EFAULT;
44     }
45
46     for (unsigned int i = 0; i < size; i++) {
47         unsigned int pos = (dgst_offset + i) % DGST_SIZE;
48         dgst[pos] = dgst[pos] ^ tmp[i];
49     }
50     dgst_offset = (dgst_offset + size) % DGST_SIZE;
51
52     kfree(tmp);
53     return size;
54 }
55
56 static struct file_operations procfs_ops = {
57     .owner = THIS_MODULE,
58     .read = read_dgst,
59     .write = write_data,
60 };
61
62 static int init_dgst(void)
63 {
64     memcpy(dgst, seed, DGST_SIZE);
65      procfs_entry = proc_create(PROCFS_NAME, 0644, 0, &procfs_ops);
66     return 0;
67 }
68
69 static void exit_dgst(void)
70 {
71     if (procfs_entry != 0) {
72         remove_proc_entry(PROCFS_NAME, 0);
73         procfs_entry = 0;
74     }
75 }
76
77 module_init(init_dgst);
78 module_exit(exit_dgst);
79
80 MODULE_LICENSE("GPL");
81 MODULE_AUTHOR("Raphael");
82 MODULE_DESCRIPTION("ProcFS Demo Kernel Module");
```

相较于表 8 的代码，我们主要新增了 11~61 行的代码，为 procfs 添加了读取摘要 (read_dgst) 与写入数据 (write_data) 的接口函数，但是没有提供打开和关闭的接口函数。

从接口函数中可以看到我们之前说的所有参数，包括文件、用户态传进来的缓存区、缓存区大小以及文件偏移量，但是没有 errno。实际上 glibc 对系统调用的返回值进行了处理，当返回值小于 0 时，glibc 会将其转换为 -1，并使用返回值的绝对值作为 errno，因此在出现缓存区访问错误时都需要返回 -EFAULT。

需要注意的是偏移量是内核模块自己维护的，可以根据偏移量的当前值判断整个文件写入到什么地方或者读取到什么地方了。在上述代码中，我们仅在读取摘要的时候设定了偏移量，并在读取摘要的函数里根据设置的偏移量判断是否已经读取过摘要了，如果读取过摘要就不能再读取，不然 cat 每次调用 read 都会有数据，从而对同一个摘要会打印无限次。

此外，一定要注意，内核不能直接访问用户

态的数据 (因为没有经过可访问性检查)，内核态从用户态复制数据可以使用 copy_from_user 或者 get_user 等，内核态向用户态复制数据可以使用 copy_to_user 或者 put_user 等，这些宏通过 linux/uaccess.h 导入即可。而内核态分配与释放内存使用 kmalloc 与 kfree，它们通过 linux/slab.h 导入即可。

在以上方法中，我们加载模块后可以通过 echo 命令向 /proc/dgst 写入数据，通过 cat 命令从 /proc/dgst 读取摘要，从而证明了我们实现的读写函数已经成功。在 read_dgst 函数里，我们没有返回实际的摘要值，而是摘要值的十六进制可打印字符串形式，主要是为了使用 cat /proc/dgst 的方法来直接看到结果方便调试。

可能已经有同学注意到了，我们实现的函数里没有 open 与 close 对应的接口，这是因为在 procfs 里这两个接口都是可以为空的，具体的实现可以参见内核代码树里 fs/proc/inode.c 与 fs/proc/generic.c 的源码。

当然，没有 open 与 close 接口在这里实际上是一个 bug，因为这样所有向此设备输出的数据都会影响到全局唯一的一个摘要。一般情况下，我们使用摘要的时候希望的是每次输入一个字符串计算出一个摘要，而不要受到之前使用过摘要函数的影响，这样，就必须实现对应的 open 与 close 函数了，参见表 11。

表 11 支持 open/close 的 procfs 模块源码

```
 1 #include <linux/kernel.h>
 2 #include <linux/module.h>
 3 #include <linux/proc_fs.h>
 4 #include <linux/uaccess.h>
 5 #include <linux/slab.h>
 6
 7 #define DGST_SIZE   8
 8 #define PROCFS_NAME "dgst"
 9
10 static struct proc_dir_entry* procfs_entry = 0;
11 static unsigned char seed[DGST_SIZE] = {0x6c, 0xb3, 0x47, 0x93, 0x21, 0x78, 0xdb, 0x2f};
12
13 struct dgst_metadata {
14     unsigned int dgst_offset;
15     unsigned char dgst[DGST_SIZE];
16 };
17
18 static int open_dgst(struct inode* si, struct file* sf)
19 {
20     struct dgst_metadata* dm = kzalloc(sizeof(struct dgst_metadata), GFP_KERNEL);
21     if (dm == 0)
22         return -ENOMEM;
23
24     memcpy(dm->dgst, seed, DGST_SIZE);
25     sf->private_data = dm;
26     return 0;
27 }
28
29 static int close_dgst(struct inode* si, struct file* sf)
30 {
31     if (sf == 0 || sf->private_data == 0)
32         return -EBADF;
33
34     kfree(sf->private_data);
35     sf->private_data = 0;
36     return 0;
37 }
38
39 static ssize_t read_dgst(struct file* sf, char __user* buf, size_t size, loff_t* offset)
40 {
41     if (sf == 0 || sf->private_data == 0)
42         return -EBADF;
43
44     if (*offset != 0)
45         return 0;
46
47     if (size < DGST_SIZE*2 + 1)
48         return -EINVAL;
49
50     struct dgst_metadata* dm = (struct dgst_metadata*)sf->private_data;
51
52     char tmp[32];
53     snprintf(tmp, sizeof tmp, "%x%x%x%x%x%x%x%x",
dm->dgst[0], dm->dgst[1], dm->dgst[2],
54             dm->dgst[3], dm->dgst[4], dm->dgst[5], dm->dgst[6], dm->dgst[7]);
55     unsigned int len = strlen(tmp)+1;
56     if (copy_to_user(buf, tmp, len) == 0) {
57         *offset = len;
58         return len;
59     }
60     return -EFAULT;
61 }
62
63 static ssize_t write_data(struct file* sf, const char __user* buf, size_t size, loff_t* offset)
64 {
65     if (sf == 0 || sf->private_data == 0)
66         return -EBADF;
67
68     if (size == 0) return 0;
69
```

```
70    char* tmp = kmalloc(size, GFP_KERNEL);
71    if (tmp == 0)
72        return –ENOMEM;
73
74    if (copy_from_user(tmp, buf, size) != 0) {
75        kfree(tmp);
76        return –EFAULT;
77    }
78
79     struct dgst_metadata* dm = (struct dgst_metadata*)
sf–>private_data;
80    for (unsigned int i = 0; i < size; i++) {
81        unsigned int pos = (dm–>dgst_offset + i) % DGST_
SIZE;
82        dm–>dgst[pos] = dm–>dgst[pos] ^ tmp[i];
83    }
84      dm–>dgst_offset = (dm–>dgst_offset + size) %
DGST_SIZE;
85    kfree(tmp);
86    return size;
87 }
88
89 static struct file_operations procfs_ops = {
90    .owner = THIS_MODULE,
91    .open = open_dgst,
92    .read = read_dgst,
93    .write = write_data,
94    .release = close_dgst,
95 };
96
97 static int init_dgst(void)
98 {
99     procfs_entry = proc_create(PROCFS_NAME, 0644, 0,
&procfs_ops);
100     return 0;
101 }
102
103 static void exit_dgst(void)
104 {
105     if (procfs_entry != 0) {
106         remove_proc_entry(PROCFS_NAME, 0);
107         procfs_entry = 0;
108     }
109 }
110
111 module_init(init_dgst);
112 module_exit(exit_dgst);
113
114 MODULE_LICENSE("GPL");
115 MODULE_AUTHOR("Raphael");
116 MODULE_DESCRIPTION("ProcFS Demo Kernel
Module");
```

从表 11 的代码可以看出我们的 open 函数主要就是通过 struct file 的 private_data 成员 (void* 类型) 保存了自己的摘要信息，然后在 read/write 函数里面就不再需要全局变量，而是访问 private_data 里面的摘要信息了，显然在这里 read/write 函数需要事先检查下 private_data 的有效性，不然应该返回 –EBADF。当然，最后在 release 函数里需要释放对应在 open 里分配的内存。

此时，我们已经无法通过 echo 与 cat 来更新查看 /proc/dgst 的数据了，因为每次 echo 与 cat 命令都会打开 /proc/dgst 然后再关闭这个文件，从而使得对文件的更改在关闭后就丢失了。因此我们需要一个用户态的测试程序，如表 12 所示。

表 12 用户态的 /proc/dgst 读写测试程序

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/stat.h>
4 #include <fcntl.h>
5 #include <unistd.h>
6 #include <string.h>
7
8 int main(int argc, char* argv[])
9 {
10    int fd = open("/proc/dgst", O_RDWR);
11    if (fd < 0) {
12        perror("open");
13        return –1;
14    }
15
16    char *buf = argc == 1 ? "hello, world" : argv[1];
17    if (write(fd, buf, strlen(buf)) <= 0) {
18        perror("write");
19        close(fd);
20        return –2;
21    }
22
23    char result[32] = {0};
24    if (read(fd, result, sizeof result) <= 0) {
25        perror("read");
26    } else {
27        printf("digest of '%s' is %s\n", buf, result);
28    }
29    close(fd);
30 }
```

此测试程序会以读写的方式 (O_RDWR) 打开 /proc/dgst 文件，先写入一段字符串，然后再读取摘要，程序相当简单，就不再赘述了。

当然，内核模块的 read_dgst 的实现使得它只能被调用一次，这样是为了方便通过 cat /proc/dgst 查看原始的摘要种子值 (seed)，不然运行 cat /proc/dgst 就会永无止境了，如果用户态程序已经考虑到

了这个问题，那就可以在 read_dgst 里把对 offset 参数的处理都丢掉了。

四、结语

Linux 内核模块涉及到的方方面面还是很多的，我们上面讲的只是一个入门简介。其实如果感兴趣，至少还可以深入了解下：

· 用户态与内核态通讯的其它方式，包括 ioctl、seq_file、sysctl、netlink、系统调用等，其中 ioctl 就很适合代替上述的 read_dgst 函数，netlink 是扩展性最好，性能最高的一种，而添加系统调用需要修改整个内核，对于内核模块来说是不可行的；

· 内核态访问用户态的信息，如当前进程的用户名、进程信息、访问文件等；

· 内核态模块之间的通讯，包括通过 EXPORT_SYMBOL 的方式直接调用其它模块的函数，或者通过 kallsyms_lookup_name 的方式通过函数名查找内核函数指针间接调用；

· 中断处理与相关的 tasklet、工作队列 (work queue) 等；

· 内核线程 (kernel thread) 与锁机制 (mutex、semaphore、spinlock 等)；

· 内核模块的调试，如使用 printk、debugfs、串口调试、kdb、kdump/crash 等方式的调试。

另外，由于内核变化很快，而内核的函数又不固定 (不像 Windows 驱动接口那样稳定)，因此很多网上的资料都过时很快。比如现在网上最常见的仍然是 2.6 时期的各种内核模块开发资料，虽然总体来说仍然是没有问题的，但是细节上改变仍然很多，举几个 2.6.32 到 3.16 以上内核变更的例子：

· 原 __devinit、__devexit 宏都消失了，会导致新内核下原代码的编译错误；

· 原内核信号量的初始化宏为 init_MUTEX 等，现在已经消失了，需要使用 sema_init；

· 原 procfs 创建节点的函数是 proc_create_read_entry 等，在新版本的内核里需要使用 proc_create，当然，也需要修改相应的参数。此外，也需要对原有的参数进行结构体封装；

· 文件系统操作结构体 (struct file_operations) 的回调函数有了更改，原来的 ioctl 改名为 unlocked_ioctl，以提高性能，对于用户态与内核态的通讯来说，使用此方法的性能将会更高，当然，函数的参数也发生了变化；

· 旧版本内核里的头文件 linux/smp_lock.h 已经被取消 (由于 BKL 已经被清除)，需要换成新版本内核的 linux/hardirq.h。

· ……

上述程序在 Deepin Server 15/15.1 与 Deepin Desktop 15.3 上编译测试通过，内核版本分别是 3.16 与 4.4，仅此提醒一下。

还有一点，由于内核接口的不稳定性，而且内核模块载入需要匹配 vermagic 的关系，内核模块还是尽量少写为好，以避免适配不同版本与分支的内核不停地编译与发布。很多任务其实不需要内核模块就可以搞定的，就不要自己开发内核模块了，比如系统参数调整，比如新的文件系统 (可以用 FUSE)、比如 USB 驱动 (可以用 libusb)。

好了，到这里了，就到这里吧，Happy Hacking, Guys ;)

五、参考

· The Linux Kernel Module Programming Guide：http://www.tldp.org/LDP/lkmpg/2.6/html/

· 黑客内核：编写属于你的第一个 Linux 内核模块：https://linux.cn/article-3251-1.html

· 在 Linux 下用户空间与内核空间数据交换的方式：https://www.ibm.com/developerworks/cn/linux/l-kerns-usrs/ https://www.ibm.com/developerworks/cn/linux/l-kerns-usrs2/

· Linux Kernel Source

Linux 生态分析

武汉 开发团队

Linux 下软件不够用，这个问题大家都明白是一个鸡与蛋的问题。另外一个比较突出的是“软件依赖”问题。

Linux 生态的问题

deepin 在刚刚完成第一个版本 DDE 后就曾试图解决这个问题，到目前为止我个人已经不再愿意去试图解决它。

从最开始思考并研究现有解决方式到现在，我越发认为“软件依赖”这个问题在目前的 Linux 生态环境下是无法解决的。因为这个完全不是技术问题，而是生态问题。

Linux 为何有如此多的发行版？ 比较片面的回答是：“Linux 不是 windows，不同发行版是把自由选择的权利留给用户”。我不否认这个回答，但我认为任何社会现象一定都是由利益驱动形成的。“自由选择”是果，不是因。

Linux 的发展目前由两个群体在推动：

1. 社区的开发者，凭着爱好或理想，为了满足自我精神追求。这个群体是 Linux 生态规则的奠基人。

2. 商业行为的开发者，由商业利益驱使。这个群体是 Linux 今日的引擎。

目前 Linux 绝大部分贡献的背后都是商业公司，然而不论是否是商业行为，大家都遵循社区最初奠定的自由精神。自由精神已经在 Linux 生态里根深蒂固。

回到“非依赖”问题上，更准确一点来说应该是期望实现以下目标：“用户可以轻松的安装任何一款合法的软件。”

解决“软件依赖”问题的方式目前有两类：

- **拆分组合方式**

将需要的组件分门别类，运行时动态组合起来，以满足软件的依赖关系。 类似 Mac 的 bundle 机制。Linux 下的代表实作为 flatpak（xdg-app），snap，这两者其实更侧重解决安全问题。

- **路径欺骗方式**

Linux 下的代表实作—— nix 。

这两条可以看到的路都已经有人实现了，大大小小的实现方案不下 10 种，也都已经能将桌面环境跑起来。并且有理论基础，有配套设施。但都没有起到任何效果，没有一个方案真正让普通桌面用户收益。

这也是我不愿意再谈及“软件非依赖”计划的原因，因为我不相信可以在技术上还能提供什么突破性的贡献。

然而“用户可以轻松的安装任何一款合法的软件”这个合理的需求怎么办？

要解决这个问题，我先解释下为何我说这个问题不是技术问题，而是生态问题。

我是 07 年开始使用 Linux 的，和大家一样，跟团旅游般，走马观花，将各种 Linux 走了一圈。 那时网上的很多教程，在涉及安装软件时，还停留在上一个阶段，即—体验非常差的手动处理包依赖。还好我接触 Linux 比较晚，那时大部分发行版已经全

部是包管理器自动处理这一堆事。 过了几年 iOS、Andorid 开始火起来，普通用户才了解到软件商店的概念。 最初几年我一直都觉得 Linux 安装软件要比其他系统方便的多。要安装什么软件只需要在命令行下 search 一下 关键字就能列出“世上所有”相关的软件以及评分 (AUR)，并且会从最近的镜像源用最优的速度帮你下载下来。完全不用去找软件，下载的时候也不用仔细甄下载链接，安装的时候也不用处处留心。除了软件不够用这个致命缺点，安装体验绝对是完爆其他平台。 但过了快 10 年 Linux 的软件生态几乎还是老样子，早已被其他平台甩开了一截，现在不仅是软件不够用这一个缺点了。

Linux 软件生态圈

在 Linux 生态圈里，面向软件这个实体角度来说，分为三类角色：

1. 软件开发者
2. 软件分发者
3. 软件使用者

与其他平台最大的不同在于“软件分发者”。这类角色就是各发行版的维护者，是他们让 Linux 越来越可用。

一个主流的发行版本大概有 3 万多个包需要维护，而一个发行版往往有多个版本多个 CPU 架构，需要维护的包在 10 万级别。

- 软件开发者：让大家都能使用上我写的软件。

软件至少要能安装、能运行。而现代软件都是构建在更底层的库之上，这些底层库逐层的封装了硬件、软件资源。底层库是一层一层搭建起来的构成最终应用软件运行的必备软件栈。

而这些软件栈就是由发行版提供的，他们大体相同，然而千差万别，不同发行版很多时候是要给用户提供不同的软件栈组合。

软件开发者往往只能测试少量几个发行版所对应的软件栈与应用软件的匹配情况。然而用户量可观的发行版少说也有 10+ 个，实际要面对的情况远远大于 10+(发行版) * 2+(架构) * 3^{+}(版本) = 60，而且这里每个维度的取值都很小而且没有算上软件发布后出现的那些运行环境。

所以软件开发者几乎无法提供一个可直接交付的软件，只能由软件分发者加工后再交付给用户。这无形中伤害了开发者的利益，现在的用户已经几乎忘记软件开发者这个角色了，因为他们面对的只有“apt-get install”。早些年 sf 上还能经常看到软件的贡献者链接，但现在发行版做的越来越好了，已经几乎没有机会去看到这些信息了。

- 软件分发者：为用户提供更多的可安装包。

这些人面对的问题更加棘手。

软件包可以抽象为一个由 (包名、版本) 组成的二维向量 (为了简化概念这里不考虑架构等实际因素)。 其中包名不会变动，但随时可能会衍射出新的版本 (软件开发者这个不确定因素在活动)。

发行版是通过软件仓库来为最终用户提供服务的。软件仓库可以抽象为由 N 个软件包组成的 N(N 大概为 3 万) 维向量 (软件包 1、软件包 2、软件包 3、软件包 N)。

大家可以想象一下 3 万多个元素一字拉开，而每个元素都会随机的上下窜动的场景。而且为增加一个元素或删除一个元素都可能导致崩溃。

发行版维护者需要不断的测试各种组合带来的影响，而可悲的是，往往没有任何一个组合能解决所有人的问题，每一种组合选择都只能解决一些问题，同时引入一些新的问题。(试想一下针对特定发行版都无法解决的问题，光靠一个新技术能解决吗)

·软件使用者：能使用上需要的软件。

得益于软件开发者的创造以及软件分发者的辛苦工作，现在 Linux 下的用户安装使用软件并没有太多的困难。相对于前两者他们的遇到的困难似乎微不足道。

虽然不同发行版提供了不同选择，但在安装软件上面，对用户来说这些方式并没有什么本质区别（非技术上），都是在做同样的一件事。这是众多发行版共同对用户造成的伤害。软件的传播并不方便，因为软件名≠包名，而 Linux 上只能安装包，而不能直接安装软件。

除了操作系统与软件协作运行导致的环境复杂性这个无法避免的原因外，还有一个非常重要的原因就是 Linux 发行版造成的软件环境“分裂”。我们这些人能用上 Linux 是因为发行版的付出，然而这种因为“自由选择”导致的环境“分裂” 也是发行版造成的恶果。

从另一个角度来看，Linux 的生态是什么样的呢?

1. 软件开发者只会考虑到有限的几种运行环境，所以只会对少量的运行环境进行支持；

2. 分发者时时刻刻都面对着数不清的需要更新的软件。毕竟永远有成堆的 bug 以及新的用户需求要靠更新这些软件来解决；

3. 使用者用着分发者见都没见过的硬件运行着他们提供的软件组合。

这个问题的解决绝对不是一个技术能解决的，首先必须有足够的硬实力，其次还要顺势而为。

第一次联手

首先我先站个队，我认为“Linux 的发展，各发行版功不可没”。

但“软件环境分裂”对大家造成的伤害（包括发行版自身），发行版需要承担很大的责任。所以 2011 年，包括 Debian、SUSE、Ubuntu、Redhat、Mandriva 等联合发起了 appstream 项目。

其核心工作就是定义了一套描述软件元数据的规范，建立起软件描述、截图，与包名的对应关系之类的数据库。appstream 不仅建立了非常完善的 spec，而且也实现了配套的基础设施库，并且 Gnome、Ubuntu 等商店已经迁移到它上面，Redhat、Debian 等仓库也对其进行了支持。

然而让人失望的是，这场持续至今的运动并没有对普通用户带来一丝影响。

真正的问题

appstream 的失败在于，这些发行版始终不敢打破“自由选择”这种“政治正确”。

appstream 技术上是完善的，在解决软件描述统一性这个问题上简直绰绰有余。然而实际运作起来，却依旧是“自由选择”每个发行版有自己的一套东西，不同的商店，不同的变种，在技术上是兼容的，然而对普通用户来说并没有什么改变，对开发者来说仅仅也只是多了一个新的软件描述方式（也就是多了一个负担）。

deepin 期望的是创造一种新的包格式以及配套的工具（开发端以及运行端）来解决这个问题，这个目前来看比 appstream 更加困难。

Linux 发行版众多是“自由选择”导致的，是根深蒂固的，我不相信其能被“解决”，因为它本身不是一个问题。 应用软件运行需要软件栈来支持，依赖问题是不可避免的，无非是谁来消化这种复杂性。

Android、iOS 靠重新打造一个封闭的独立新生态，将问题的出现拖延并缓解了。

Windows 靠封闭以及强大的话语权将软件环境的变动减少到最小，即使有变动也是计划性变动，会有准备周期。

而从最开始 Linux 在这个问题上就没有什么规划或限制，导致如今已经无法单纯通过技术来解决这个问题。 尝试取代 deb 或 rpm 系无非是再创造了一个 PKGBUILD 系。

如何解决

这种无奈的格局目前是无法打破的，但我们可以做整合资源，而且其代价非常小。

因为这个问题里最困难的部分，软件组合复杂性，每个发行版都已经控制住了。不论你使用 archlinux、ubuntu 还是 deepin 只要按照官方提供的方式基本不会出现什么大问题。

发行版最基本的是需要独立人格，其中很大一部分是由这些软件栈的组合来体现的，并且需要持续不断的完善这些软件栈。

那我们还要做什么呢？我觉得需要分别从 3 个方向来看待：

1. 给用户提供统一友好的界面（普通用户需要的是统一友好的预期行为）；

2. 在尊重不同发行版的原则的前提下减少其对用户的负面影响（同时提供有价值的反馈信息）；

3. 拉近开发者与最终用户的距离（开发者需要更多的用户影响力）。

只有让 Linux 生态中重要的 3 个角色都能获益才能进入一个正反馈的循环。

具体要做的事有：

1. 分离深度商店软件与深度软件仓库，建立一个对应关系数据，并维护其他发行版的对应关系；

2. 以深度商店为原型，逐步移植到各大发行版，并建立软件包与商店软件的对应关系；

3. 在商店中提供更多软件作者的信息，如项目主页、作者信息、软件捐款链接等；

4. 培养用户习惯，建立用户习惯→软件→深度商店的一个循环。但这个前提一定是开发者以及普通用户切实收益于深度商店才可能实现；

5. 提供更换的软件管理体验，比如软件安装情况云端记录；

6. 提供更完整的软件数据。所有 Linux 的软件不论是否在仓库中都应该出现在软件商店中。数据完整不是错，为了减少信息杂乱对用户的 不良干扰，完全可以通过交互设计来隔离。

总结来说，努力给用户更好的体验，努力给开发者更多的宣传，尽力尊重各发行版的原则。 只有这样才有可能在另外一个层面上建立起一个统一的环境，才有进行真正技术革新的硬实力。

浅谈 Linux 下功能强大的网络配置工具 ——ip

◉ 北京 研发团队

深度服务器操作系统中管理员用户可使用简单实用的 ip 工具程序对 ip 进行管理，它也是 Linux 下较新的功能强大的网络配置工具。

一、ip 工具程序简介

ip 命令用来显示或操纵 Linux 主机的路由、网络设备、策略路由和隧道，是 Linux 下较新的功能强大的网络配置工具。特别的，不赞成使用的 linux 网络命令有 arp, ifconfig, iptunnel,nameif, netstat, route。这些程序包含在 net-tools 软件包，但该软件包已经多年不被维护了。他们中的许多基本功能能够用 iproute2 软件包中一些套件替代，其中最著名的就是新的 ip 命令。下面我们先介绍 ip 的语法，再介绍 ip 与上述命令的对比。

二、ip 的一般用法

ip [OPTIONS] OBJECT [COMMAND l help]

其中，OPTIONS 是一些修改 ip 行为或者改变其输出的选项，其常用参数请参见表 1.4。

表 1.1 ip 命令 OPTION 选项参数说明

短参数	长参数	参数说明
—V	—version	显示指令版本信息
—s	—stats，—statistics	输出更详细的信息
—f	—family	强制使用指定的协议族
—4		指定使用的网络层协议是 IPv4 协议
—6		指定使用的网络层协议是 IPv6 协议
—0		输出信息每条记录输出一行，即使内容较多也不换行显示
—r	—resolve	显示主机时，不使用 IP 地址，而使用主机的域名。

而 OBJECT 是要管理或者获取信息的对象，其常用对象请参见表 1.5。

表 1.2 ip 命令认识的对象

短参数	长参数	参数说明
address		一个设备的协议（IP 或 IPv6）地址
link		网络设备
route		路由表条目
rule		路由策略数据库中的规则
neighbour		ARP 或者 NDISC 缓冲区条目

三、网络命令对比

表 1.3 网络管理的其他命令与 ip 命令对比

其他网络命令	等效的 ip 命令
arp	ip n (ip neighbor)
ifconfig	ip a (ip addr)
iptunnel	ip tunnel
nameif	ip link
netstat	ss，ip route (for netstat—r)，ip —s link (for netstat —i)，ip maddr (for netstat —g)
route	ip r (ip route)

表 1.4 arp vs ip

arp 命令	等效的 ip 命令
arp —a [host] or ——all [host] 显示指定主机或指定 ip 地址的条目	ip n (or ip neighbor)，or ip n show
arp —d [ip_addr] or ——delete [ip_addr] 移除指定主机的 arp 缓存条目	ip n del [ip_addr] (this “invalidates” neighbor entries) ip n f [ip_addr] (or ip n flush [ip_addr])

arp —i [int] or ——device [int] e.g. arp —i eth0 —s 10.21.31.41 A321.ABCF.321A creates a static ARP entry associating IP address 10.21.31.41 with MAC address A321.ABCF.321A on eth0.	ip n [add | chg | del | repl] dev [name]
arp —s [ip_addr] [hw_addr] or ——set [ip_addr] 创建一个静态的 arp 地址对指定的 ip 主机	ip n add [ip_addr] lladdr [mac_address] dev [device] nud [nud_state]（见下面实例）
arp —v 使用 verbose 模式显示更多细节	ip —s n (or ip —stats n)

举例如下：

```
# ip n del 10.1.2.3 dev eth0
```

Invalidates the ARP cache entry for host 10.1.2.3 on device eth0.

```
# ip neighbor show dev eth0
```

Shows the ARP cache for interface eth0.

```
# ip n add 10.1.2.3 lladdr 1:2:3:4:5:6 dev eth0 nud perm
```

Adds a “permanent” ARP cache entry for host 10.1.2.3 device eth0.

表 1.5 ifconfig vs ip

ifconfig 命令	等效的 ip 命令
ifconfig	ip a (or ip addr)
ifconfig [interface]	ip a show dev [interface]
ifconfig [address_family]	ip —f [family] a
Ifconfig [interface] add [address/prefixlength Adds an IPv6 address to the [interface].	ip a add [ip_addr/mask] dev [interface]
ifconfig [interface] address [address]	ip a add [ip_addr/mask] dev [interface]
ifconfig [interface] allmulti or —allmulti	ip mr iif [name] or ip mroute iif [name]
ifconfig [interface] arp or —arp	ip link set arp on or arp off
ifconfig [interface] broadcast [address] .	ip a add broadcast [ip_address] ip link set dev [interface] broadcast [mac_address]
ifconfig [interface] del [address/prefixlength]	ip a del [ipv6_addr or ipv4_addr] dev [interface]
ifconfig [interface] down	ip link set dev [interface] address [mac_addr]
ifconfig [interface] mtu [n]	ip link set dev [interface] mtu [n]
ifconfig [interface] multicast	ip link set dev [interface] multicast on or off
ifconfig [interface] promisc or —promisc	ip link set dev [interface] promisc on or off
ifconfig [interface] txquelen [n] Sets the transmit queue length on the [interface]. Smaller values are recommended for connections with high latency (i.e., dial—up modems, ISDN, etc).	ip link set dev [interface] txqueuelen [n] or txqlen [n]
ifconfig [interface] tunnel [address]	ip tunnel mode sit
ifconfig [interface] up	ip link set [interface] up

ip 命令用法实例如下：

```
# ip link show dev eth0
# ip a add 10.11.12.13/8 dev eth0
# ip link set dev eth0 up
# ip link set dev eth0 mtu 1500
# ip link set dev eth0 address 00:70:b7:d6:cd:ef
```

表 1.6 iptunnel vs ip tunnel

iptunnel 命令	等效的 ip 命令
iptunnel [add | change | del | show]	ip tunnel a or add ip tunnel chg or change ip tunnel d or del ip tunnel ls or show iptunnel add [name] [mode {ipip | gre | sit}]
remote [remote_addr] local [local_addr] remote [remote_addr] local [local_addr]	ip tunnel add [name] [mode {ipip | gre | sit | isatap | ip6in6 | ipip6 | any }]

Iptunnel 和 ip tunnel 用法实例如下

```
# [iptunnel l ip tunnel] add ipip-tunl1 mode ipip remote 83.240.67.86 (ipip-tunl1 is the name of the tunnel, 83.240.67.86 is the IP address of the remote endpoint).

# [iptunnel l ip tunnel] add ipi-tunl2 mode ipip remote 104.137.4.160 local 104.137.4.150 ttl 1

# [iptunnel l ip tunnel] add gre-tunl1 mode gre remote 192.168.22.17 local 192.168.10.21 ttl 255
```

表 1.7 nameif vs ip

Nameif 命令	等效的 ip 命令
nameif [name] [mac_address] ifrename -i [interface] -n [newname]	ip link set dev [interface] name [name].

表 1.8 netstat vs ss/ip

netstat 命令	等效的 ss 命令
netstat -a or --all	ss -a or --all
netstat -A [family] or --protocol=[family]	ss -f [family] or --family=[family]
netstat -C	ip route list cache
netstat -e or --extend	ss -e or --extended
netstat -g or --groups	ip maddr, ip maddr show [interface]
netstat -i or --interface=[name]	ip -s link
netstat -l or --listening	ss -l or --listening
netstat -n or --numeric	ss -n or --numeric
netstat -N or --symbolic	ss -r or --resolve
netstat -o or --timers	ss -o or --options
netstat -p or --program	ss -p
netstat -r or --route	ip route, ip route show all
netstat -s or --statistics	ss -s
netstat -t or --tcp	ss -t or --tcp
netstat -u or --udp	ss -u or --udp
netstat -w or --raw	ss -w or --raw

表 1.9 route vs ip

route 命令	等效的 ip 命令
route	ip route
route -A [family] [add] or route --[family] [add]	ip -f [family] route
route -e or -ee	ip route show
route -h or --help	ip route help
route -v or --verbose	ip -s route
route -V or --version	ip -V
route add or del	ip route [add | chg | repl | del] [ip_addr] via [ip_addr]
route [add or del] dev [interface]	ip route [add | chg | repl | del] dev [interface]
route [add or del] [default] gw [gw]	ip route add default via [gw]
route [add or del] metric [n]	ip route [add | chg | repl | del] metric [number] or preference [number]
route [add or del] mss [bytes]	ip route [add | chg | repl | del] advmss [number]
route [add or del] reject	ip route add prohibit [network_addr]
route [add or del] window [n]	ip route [add | chg | repl | del] window [W]

Ip route 命令语法实例如下：

```
# ip route add 10.23.30.0/24 via 192.168.8.50
# ip route del 10.28.0.0/16 via 192.168.10.50 dev eth0
# ip route chg default via 192.168.25.110 dev eth1
# ip route get [ip_address]
```

四、Ip 网络配置工具应用举例

1. 显示网络设备及配置

```
ifconfig
ip addr show
ip link show
ip addr show dev eth0
```

2. 开启和关闭一个网络接口

```
ifconfig eth0 up
ip link set eth0 up
ifconfig eth0 down
ip link set eth0 down
```

3. 设置 ip 地址

```
ifconfig eth0 192.168.0.77
ip address add 192.168.0.77 dev eth0
ifconfig eth0 192.168.0.77
netmask 255.255.255.0 broadcast
192.168.0.255
ip addr add 192.168.0.77/24 broadcast
192.168.0.255 dev eth0
```

4. 删除一个 ip 地址

```
ip addr del 192.168.0.77/24 dev eth0
```

5. 给网络接口添加一个别名

```
ifconfig eth0:1 10.0.0.1/8
ip addr add 10.0.0.1/8 dev eth0 label eth0:1
```

6. Arp 协议

在 arp 表中添加一条

```
arp –i eth0 –s 192.168.0.1 00:11:22:33:44:55
ip neigh add 192.168.0.1 lladdr
00:11:22:33:44:55 nud
permanent
dev eth0
在一个设备上关掉 arp 方案
ifconfig –arp eth0
ip link set dev eth0 arp off
```

7. 显示路由信息

```
ip ro
```

8. 添加和删除路由

```
ip ro add 10.0.0.0/16 via 192.0.2.253
ip ro del 10.0.0.0/16 via 192.0.2.253
```

9. 获取 arp 表

```
ip neigh
```

10. 获取 ip neighbor 的帮助

```
ip neighbor help
```

深度 · 讲坛 征稿啦

是什么让你豁然开朗？是什么让你灵感迸发？又是什么让你百思不解？

这里汇聚了深度大神，专门凑在这里搞技术，从今往后，这里专治各种疑难杂症，你可以来提问，可以来传授技术，总之，这里都是爱凑热闹的技术控。

分享专业知识，方便学习交流，内刊“深度 · 讲坛”栏目长期向各位技术控们征集技术稿件啦。

字　　数：1000+，图文并茂
内　　容：必须为原创
投稿邮箱：deepin–magazine@linuxdeepin.com
　　注明“部门 + 姓名”

陪伴是最长情的告白

武汉 王勇 / 文

妹妹春节来武汉玩，作为已经在武汉生活了 6 年的“土著”，自然要担当“导游”。其实武汉真的没什么好玩的，特别是春节，整个武汉就像外国人眼中的空城一样，全方位萧条。

但是对于第一次来武汉的游客来说，还是有几个经典景点值得一游的。

因为春节放假，武汉的一些小资情调的咖啡厅、小店啊都关门了，这次春节武汉游主要就是去那些不会因为节日关门的地方压马路。

第一日：户部巷

来武汉必去的景点就是吃货街——户部巷，我个人对户部巷的评价就是，如果你没有太多时间的话，可以去户部巷一条街去吃武汉比较经典的小吃：热干面、豆皮、扇贝、小龙虾；如果时间充裕，建议去武汉街边去吃，便宜又实惠，而且不用在人堆里面挤来挤去。

游前做攻略，旅游不会累。去任何景点之前，可以找一些小景点，然后沿途步行，更能体验当地的自然风光和风土人情，旅游是要用心体验脚下的每一步，而不是盲目地拿着相机咔嚓咔嚓和自拍，一定不能忽略真正沁人心脾的微风和心境。

上图就是我第一日的简单的规划，在马蜂窝或询问当地人大小景点，然后再配合高德地图，10 分钟搞定。

首先，从光谷广场坐地铁到首义路站（离黄鹤楼最近的地铁站）下车，出了首义路地铁站就有一个卖大芒果的小摊，老板人很好，在削了一个生芒果以后又给我们换了一个，看着金灿灿的芒果在阳光下流淌芒果汁，我吃货的小宇宙瞬间爆发啦，哈哈。

首义路旁边就是首义广场，首义广场传说是辛亥革命的起义地点，由于我是“自然景观控”，加上今天的主要目的是吃，所以就没有去辛亥革命博物馆参观，“历史控”可以去里面看看。

首义广场的林荫小道，旁边有很多小鸟飞来飞去，空气夹揉着小草的清新气息，我想这才是大武汉最让人眷恋的味道，而不是灯红酒绿的嬉闹，也不是商业街的人头攒动，是这种大爷大妈们每天留下的生活气息的再古朴不过的买菜小道，和踏在凹凸不平的林荫小道上的那种踏实感……

穿过首义广场就踏进黄鹤楼景区的范围了，特别是到了秋天，街边的银杏树叶落了一地，当清晨的阳光洒在这些金黄的叶子上时，简直美爆了。可惜此时是冬季，银杏叶都落完了，补一张去年春天的照片吧。

黄鹤楼其实是不建议各位去看的，因为当年的黄鹤楼早已被长江水淹没了，现在的黄鹤楼都是建国以后重建的，连位置都换了，而且楼里面真的就是一座空楼，真的是性价比超低的景点。武汉当地有一句玩笑话：“不去黄鹤楼很后悔，去了更后悔……”

顺着黄鹤楼景区的人行道朝长江大桥走去，中间会有一条商业街，平常都是人满为患的外地人，但是这次是春节期间，人超级少，逛着街边的小饰品店别有一番风味呢。

逛完以后再走 500 米就到黄鹤楼景区的后门，如果真想拍照留恋，后门就可以看到整座黄鹤楼。（但是千万千万不要进去哈。）

从黄鹤楼后门往前走 100 米就是长江大桥，顺着右边的

小道下桥右拐走 500 米就是今天的目的地——户部巷。

户部巷，除了热干面、豆皮、扇贝、小龙虾（夏天才有）外，其他都不是武汉本地小吃，主要都是什么肉串、螃蟹、长沙热豆腐这种全国通用景点小吃。因为在武汉生活太久，实在没有太多食欲，看官们可以自行去网上找户部巷的美食图瞅瞅。

吃完户部巷以后，就带着妹妹徒步游览长江大桥。传说这是中国第一座复线铁路、公路的两用桥，长江大桥全长 1.2 公里，半个小时漫步即可走完。走在上面会有点小恐怖，特别是当你从桥边往下望，同时背后有公交车飞驰而过的时候，桥面随着公交车舞动，真的感觉桥随时都有崩塌的可能，这也许是我天生恐高吧。

压完长江大桥马路后，旁边就是一个小景点——晴川阁，这个景点不用门票，只用在门口登记就可以进去，主要就是一个小庙宇，非常清净和惬意。

走在晴川阁的小道上，一抹夕阳洒下，让人好想好好躺着晒一下太阳，可惜此时已经饥肠辘辘，要赶紧赶路到江对面的江汉路（汉口最繁华的商业街）觅食。餐后本来准备坐晚间轮渡回武昌的，但是想了想轮渡以后还要徒步，一行人此时已经筋疲力尽，最后只能选择乘坐地铁回府。

第二日：东湖绿道

第二天一大早就赶到东湖磨山，开始东湖绿道的一天骑行游玩。

东湖绿道是横跨在东湖（中国最大的市内湖）的小道，全程没有机动车，只限自行车通行，简直是骑车者和长跑者的天堂，路两边就是东湖美景，空气好到爆。

湖中间有一块爱琴岛，把自行车仍在草坪后坐在湖中石上，再仰望四周，忘却了都市的喧闹，只管享受微风吹过的宁静……







骑行了一下午以后，坐着农家渔船返回湖对面的武汉大学，船长大爷是一个非常开朗的“顽童”，除了给我们介绍沿途景点，还在船上一直说“小伙子，你要把景拉进来”，简直就是一场东湖摄影培训，哈哈哈。

从东湖回到光谷广场，在世界城吃了海底捞（每次逢节必去，就是为了体验服务）后，带着妹妹逛了光谷广场的“绿野仙踪”和“星空走廊“，就在意大利风情街和西班牙风情街的空中走廊。特别是“星空走廊”，平常好多人都来这里摆 Pose 拍照，还有拍婚纱照的，照片已经无法形容星空走廊的美了。

第三日：年夜饭

大年三十哪儿也不去，就在家做年夜饭吃、品评春晚。

第四日：欢乐谷

最后一天，早上看了电影后，就去逛今年的欢乐谷夜场——梦幻奇妙灯会（门票 100），晚上去逛实在太美了，什么都不说，大家看图感受吧，哈哈。

一晚上坐了无数次旋转木马和碰碰车，只是不敢坐云霄飞车和任何让我眩晕的玩具。

陪伴是最长情的告白

四天的时光很快就过去了，妹妹早上送你去车站真的有点不舍呢，希望你在大都市天天开心，生活幸福。

哥哥在此帮你比武招亲啦，哈哈哈哈 ....

ATM 国产化之我见

文雷 / 文

ATM 在我国仍处于高速发展期，最近几年的年出货量超过了 20 万台。我国现在 ATM 机普遍使用的操作系统是 Windows XP，遵循的业务规范是 CEN/XFS，即欧洲标准委员会制定的金融扩展服务规范。而 CEN/XFS 本身是微软基于其自身 WOSA 架构提出来，原名为 WOSA–XFS，即基于 WOSA 的 eXtensions for Financial Services，之后移交给欧洲标准委员会的，因此此规范本身与 Windows 操作系统有很深的绑定。现在微软已经宣布对 Windows XP 停服，我国政府也已经宣布不再采购 Windows 8(Windows 10 中国定制版可以使用)，因此从长期着眼，ATM 机需要使用国产操作系统，以增强安全可控性。

就个人认为，ATM 系统的国产化可以大体分为下面几步：

· 在实验真机上实施国产化，主要通过国产操作系统厂商的努力，将现有的基于 Windows 的，遵循 CEN/XFS 规范的 ATM 业务系统运行在国产操作系统上，支持查询、存款、取款、汇款等常用功能；

· 在 CEN/XFS 的基础上，尽量保持兼容性，制定中国的实验性金融服务规范，并根据此规范，小范围选定银行、国产操作系统厂商以及 ATM 机厂商，根据此实验性规范开发出真机 ATM，在此 ATM 机上将运行国产操作系统以及基于国产操作系统与实验性规范开发出来的新的 ATM 业务系统，并能支持查询、存款、取款、汇款等常用功能；

· 由金融主管机构牵头，由中国各家银行、各主流 ATM 机厂商、各主流国产操作系统厂商建立产业联盟 (或基金会，或技术委员会)，维护与发展正式的金融服务规范，并根据规范开发公用的基础框架、开发库、测试工具等，进行技术培训，操作系统剪裁、优化与安全加固，形成完整的产业群，建立从规范到操作系统，直到业务系统的 ATM 机系统，更长远看来，还可以加入国产硬件 (芯片、内存、循环机芯等)，达成真正的全国产化，以及整体的安全可控自主化；

在第一步，深度科技的 DeepinWine 技术可以起到引导和支撑作用。它是深度科技推出的一个 Windows 兼容层方案，其目的是让 Windows 程序可以不经过修改直接运行在基于 Linux 内核的深度操作系统上，其技术原理是利用 Linux 系统调用实现了数万个 Windows API，并将其封装成 Windows 动态链接库的形式，通过 DeepinWine 载入运行 Windows 程序的时候，Windows 程序其实使用的是这些 Windows API 为表，Linux 代码为里的动态链接库，这样就实现了 Windows 程序在深度操作系统上顺利运行的目的。

它的优点在于基本不需要原应用软件厂商的支持，也不需要修改原应用软件的任何代码，即可顺

利迁移原 Windows 应用软件，而且性能基本保持不变。它的缺点是无法迁移用户自行开发的 Windows 驱动程序 (除了深度科技已经支持的部分 USB 设备与打印设备之外)，而且在应用软件很复杂，依赖 Windows 复杂功能的时候，可能需要额外的时间来适配，这是因为微软的操作系统并不开源，因此深度科技在实现 DeepinWine 的时候可能会有少部分代码尚没有 Windows 完全兼容导致的。此外，它只是一个过渡方案，长期来看，我们不可能采用在 Windows 平台上开发，在 Linux 平台上实施运营的方案。因此从长远看，我们仍然需要设定新的金融服务规范，并依此规范重新开发 ATM 业务系统。

在之前的国产操作系统迁移过程中，深度科技广泛使用了 DeepinWine 技术，对东方通信公司 ATM 机上的 ATMC、SP 管理进程、XFS 管理器、VDM 进程等进行了迁移，最终构建了一个基于原有硬件环境与 ATM 机业务系统，运行在深度操作系统上的 ATM 机系统，并支持查询、存款、取款、汇款等常用功能。系统运行稳定，功能完整，性能优良。

为了保持现有 ATM 系统迁移的速度与稳定，在开始制定 ATM 系统中国金融服务规范的时候，我们需要基于我国 ATM 系统的主流规范 CEN/XFS 进行改造，其改造原则应该基于以下几点：

· 将 CEN/XFS 里与 Windows 相关的技术方案，包括 Windows 消息、注册表、文件映射、路径等都加以修订，使用 Linux 下相应的机制实现同样的功能；

· 尽量保证原 CEN/XFS 规范中的接口、参数、功能、返回值等不发生变化，这些接口包括 API、SPI、支撑函数，以及各设备类的规范，为了加快验证迁移过程，可以先仅制定少量设备类的新规范；

· 需要加入我国金融行业的规范要求，包括密码键盘使用国密算法，以及出钞入钞遵循人行的冠字号管理方法等；

· 使用新的名称、数据类型定义以及命名规范修改原规范，使之符合 Linux 操作系统的一般开发标准；

在具体技术替代方案上，可以采用 D-Bus 替代原 Windows 消息，采用 json 数据库或者 INI 文件代替原 Windows 注册表，采用 Linux 共享内存机制代替 Windows 文件映射机制。

初始的金融服务规范制定之后，需要集中银行、主流国产 ATM 厂商，以及关键国产操作系统厂商进行开发，在新规范与国产操作系统的基础上开发出纯 Linux 的 ATM 业务系

统。此业务系统的主要目的是验证新规范、新系统的可行性，因此系统不会有大量的优化，主要着眼与正常 ATM 业务功能的实现。整个开发过程将分为以下几个阶段：

1. 独立开发，包括操作系统厂商开发管理器与测试工具，ATM 机厂商开发 SP 与 ATMC 等业务系统，硬件厂商开发 Linux 驱动

2. 集成测试，各厂商开发的软件系统按照规范整合在一起集成成为完整的 ATM 系统，我们认为这是整个过程中最耗时的部分

3. 系统测试，整机上线测试

初始的金融服务规范由于为了尽量保持与 CEN/XFS 的兼容性，肯定没有充分利用 Linux 操作系统的特性，会有极大的优化余地，因此在初始版本的金融服务规范得到验证实现之后，我们马上就需要根据开发过程中获取的经验指定正式的金融服务规范，并对其进行真正的优化，包括规范的优化，以及操作系统的裁剪指导原则，其主要目标是建立整机一体化的规范，力争使得新的 ATM 整机系统在功能、安全与性能上都超越原有系统。

在其他相关系统上，迁移方案可以与上述路径基本一致，就是首先使用类似 DeepinWine 技术进行实验性验证，在客户与业务系统厂商担心迁移风险的时候由操作系统厂商承担主要的迁移工作，对现有的业务系统进行平滑迁移。其次，在达成了实验性产品验证的目的，或者可以直接跳过前一阶段的时候，操作系统厂商，如深度科技再针对具体业务与客户以及业务系统厂商协作，制定新的开发规范，提供底层操作系统，并承担技术培训、操作系统剪裁、优化、加固的工作，以帮助客户快速、平

滑完成业务系统的迁移。

在 ATM 系统常见的基于 HTML 页面系统的迁移方面，个人推荐业务系统开发商使用 Qt 内嵌 Webkit 渲染引擎 (现在 WebEngine 尚未成熟)，通过底层的 QObject 向上层的 javascript 暴露高权限接口的方式进行迁移，同时底层的 QObject 接受 D-Bus 消息，调用上层的 javascript 函数以接口。

我们认为，为了保证产业规范的公正中立与可维护行，需要建立一个联盟 (或基金会，或技术委员会)，由金融主管机构牵头，由中国各家银行、各主流国产操作系统厂商、各主流 ATM 机厂商参与，通过会员会费的机制，由联盟维护产业规范的发展、标准框架的实现以及标准的测试与维护工具集，同时制定操作系统的裁剪标准，以保证整体系统的最优化与安全性，达成真正的自主可控、可持续发展的战略目标。

此外，我们认为，ATM 国产化或说自主可控除了安全与合规性要求以外，还有其商业上现实的目的：

· Windows 系统最近几年变化极大，例如 Windows XP 不再维护，Windows 8 不能纳入政府采购，Windows 10 采用强制滚动更新，Windows 市场份额有所下降等。Windows 的变化使得传统上依赖 Windows 的金融自助设备面临较大的维护风险，使用 Linux 平台可以有效降低此风险

· Windows 的可裁剪性较差，包括安全、性能等方面无法太过深入，也使得相应 ATM 机的可定制性与安全性较差，从而提升了成本

· ATM 机总体来说功能更新较慢，银行普遍希望降低其采购价格。对于 ATM 厂商而言，采用其他架构，例如 ARM 就是一个性价比较高的选择，但是 Windows 对异种架构的支持非常有限，同样，定制也非常困难，但是 Linux 要达到这个要求比较简单

· XFS 标准并不符合我国 ATM 机的实践，我国行业与厂商话语权小。其中部分规范从未被实现，而且没有纳入我国相应的标准，比如上述提及的冠字号规定与国密算法标准

· 由于 Linux 的灵活性与技术性操作系统厂商对 Linux 的熟谙，相关厂商为 ATM 厂商定制的操作系统能在同样的硬件平台上提供同样或更优的功能与性能，并且价格低于 Windows，可以降低 ATM 的总体成本

综上，我们认为 ATM 国产化是一条不仅从长远看来更安全可控，也是从长远看来性价比更高的目标，只要采取分步实施的方法就可以有效地降低风险，促进整个产业更好的发展。

Linux 的成长史

阮济生 / 文

我是一个小白，曾经是，现在依然是。20150822 是我第一次使用 deepin 的日期，而我的第一台笔记本却是在 20150812 号到达的，那也是我的第一台电脑，因为刚刚考上大学，嘻嘻……

其实我在 2014 年的时候就已经开始关注 linux 了，我记得我是因为看到一篇关于 Android 的报道，对对……我想起来了，好像是甲骨文和谷歌的官司，关于 Android 滥用 Java 的问题。我一直都是一个 IT 爱好者，但是我那个时候并不了解 Android，因为我那个时候的手机还是一台诺基亚，但是我周围的同学很多都已经在用那些大屏的手机了，只不过还没有现在这么大。我当时看到那个新闻后，就很好奇这个手机系统到底是何来头，让智能手机遍地开花。于是我就 Baidu 了一下，发现这是一款基于 Linux 开发的智能手机系统，谷歌一直都是免费授权给手机制造商们使用的，所以造就了这个系统的普及。我当时立刻就崇拜上了 Google（现在的 alphabet），发现这个公司好伟大，竟然可以免费给我们使用系统，但是极客精神让我觉得我不能停止，Android 系统在智能手机上的表现不错，我就很好奇它的老爸 Linux 到底是怎样的呢?

然后，我又百度了一下……从此，我的 Linux 坑路开始了。

Linux 是林纳斯·托瓦兹 这位神奇的先生发明的，而那个时候他还是名大学生，而且初衷只是为了好玩……写了 Linux 的基本代码，然后将其放在了互联网上，供其他人免费使用，连他自己都没料到，他的“好玩”造就了一个伟大的系统。这款系统后来经过一位又一位的程序员增砖添瓦，楼宇渐成，让后来世界上的很多人都注意到了这款免费的系统，而且托瓦兹先生一直都没有对 Linux 进行收费。

我当时看到了这些文字后，我的内心是激荡的，我真的被这位托瓦兹先生的魅力吸引了，加上我的内心一直有着一颗“极客梦”，从那开始，我一直在关注 Linux 方面的新闻。生活总是这样，充满各种各样的可能，谁能想到起初的一篇新闻，后来演变成了一个年轻人对 Linux 的崇拜。

我一直期待着等我拥有电脑的那一刻，也一直

林纳斯·托瓦兹

期待着 Linux 在我手中的模样。

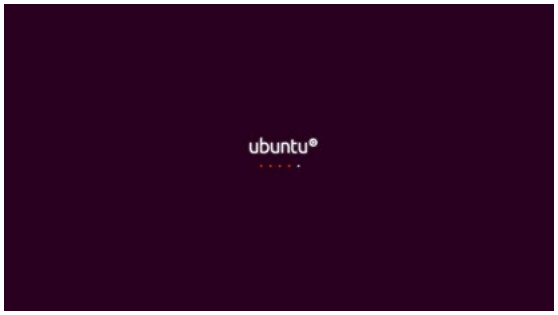
笔记本入手后，我就迫不及待的开机了，映入眼帘的是微软的标志，原装系统是 Win8，这款系统应该是 Windows 家族最短命的系统了，我哥给我装了 Win7（那个时候我哥在我的眼中可是一位电脑达人），电脑装好 Win7 后，我就开始小心又喜悦的开始使用电脑了，一开始我并没有安装 Linux 系统，因为我知道 Linux 的学习过程不像 Windows 那么简单，而且网上的很多教程都说要用虚拟机学习。

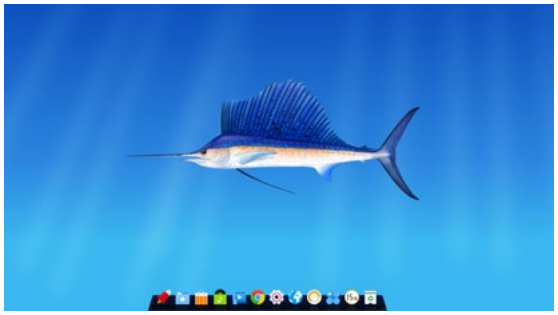
我在电脑上安装了 VMware station（免费版），然后照着网上的教程安装了一款广为人知的初学者 Linux——Ubuntu。那个时候 Ubuntu 有两个发行版，15.10 与 14.04 。我用的是 15.10，这是一个开发版，当时 Ubuntu 最新的功能都在里面了。虚拟机的安装很简单，仅用了一会儿就安装成功，然后重启虚拟机，开始体验 Linux 的魅力。

在虚拟机中出现 Ubuntu 的标志，紫色的背景，然后铛铛……进入了主界面，输入密码即可进入桌面，第一次亲自使用 Linux，我的内心还是很激动的，手指都有微微的颤抖，密码我记得都输错了三次。打开桌面，是一个与众不同的操作界面，就随便打开了 Firefox 体验，发现跟 Windows 没有什么差别，而且因为 Linux 的小巧，系统的内存占用率一直不高，同时打开 Firefox 和 libreoffice 都不会有卡顿。

随着安装 Linux 成功，我知道 Linux 最牛的是“终端”，我于是就上网开始搜索一些 Linux 论坛，想学习一些简单的命令，当时还不知道《鸟哥的 Linux 私房菜》（要是想学 Linux，我推荐这本书），在搜索 Linux 论坛的时候，我发现国内的 Ubuntu 论坛好像都不怎么活跃，瞬间浇灭了我对 Linux 的热情。正所谓：“山重水复疑无路，柳暗花明又一村。”，我无意间看到有人说 deepin 论坛很热闹，是的，deepin 论坛很热闹，你可以发现原来 Linux 用户有这么多啊。

我加入了 deepin 的家庭，注册了论坛账号，然后，我就用上了 deepin15.2 ，在实体机安装的，没有任何犹豫，看到了它的截图，看到了它的社区，看到了热情。到现在我都一直秉持着这样一个观点——玩 Linux 需要一个好的社区。在中国，用 Linux 的人太少，网上关于 Linux 的解决方案都有点老了，如果不依靠一个好的、有活力的社区，你的学习之路可能会很艰难，使用的道路也会很坎坷。





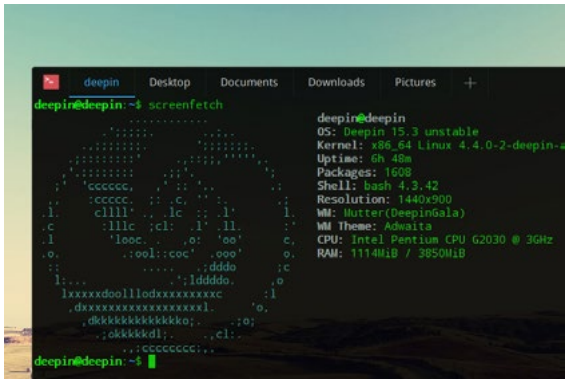
deepin 的开机很快，支持 uefi 模式，只是在 15.2 时期，还有很多东西都不是自己的，毕竟是国产系统，年龄还太小（相对于 Ubuntu、SUSE、Redhat），美化的还不错，很漂亮，那个时候论坛就有人说——好像苹果。终端也是仿照 gnome 风格的。那个时候真是小白，什么都不会，安装系统都是使用默认方式安装。每天看到论坛那些会代码的人，心里都是羡慕与嫉妒的，系统一出现问题就去论坛询问，deepin 论坛在我的 Linux 生活初期真的给了我家一样的感觉。那个时候刚好是暑假，每天我都逛论坛，学到了如何使用 sudo 提升权限，如何用 apt-get 安装软件，如何卸载软件，如何清理多余的依赖……我后来也学会了使用 man 和 wiki。

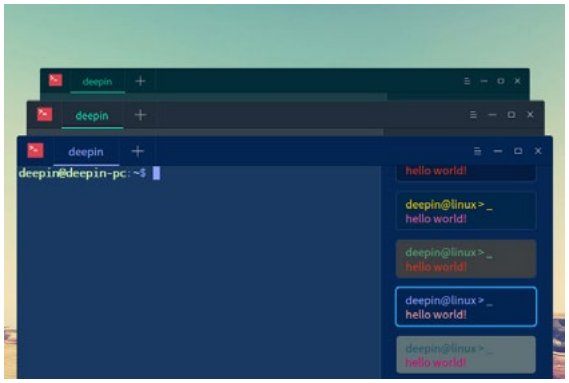
在我逐渐深入学习 Linux 的时候，我逐渐了解了一些关于电脑的处理机制、工作原理。Linux 不同于 Windows，在 Windows 中，我们的很多动作都被埋在了鼠标与美丽的 ui 下。在 Linux 的世界里，你的所作所为都是可以看到的，你可以对自己的电脑做任何事情。因为一切皆文件。

我忘了 deepin 是什么时候从 15.2 升到 15.3 的了，但是不可否认的是，deepin 成长了好多，在 15.3 的世界里，我好像看到了一个更加有自我意识的人，deepin 的自我特色愈加的明显了，开始研发很多 deepin 自主软件（那个时期应该在研发 deepin 文件管理器）。就好像看着一个小孩子，从咿呀学语开始，现在终于讲出了第一个词，那个时候我刚好在陪我满 8 个月的小侄女玩，所以深有感触，她的嘴里总是吐着一些音节。

就如同小孩子刚开始学习走路一样，我使用 Linux 的时候，并不总是一帆风顺的，我在那期间也是遇到了很多问题，例如 NTFS 无法正常挂载，图形界面崩溃……但是这些都在论坛里通过提问或者检索得到了解决，当然有些问题就是系统的 bug，官方人员也会及时进行处理的，论坛的几位管理者还是很尽责的。我在论坛里认识了几位挺厉害的 Linux 使用者，他们教会了我很多，让我知道了开源的意义，也让我知道了一些开源协议，不仅仅有 GPL 哦。

15.3 也迎来了数次更新，deepin 现在的终端真漂亮，我想它应该是最美的终端了。在那样的终端里敲着东西，你会感觉骄傲，这是一款国产系统，这个国产系统的技术似乎在逐渐超越一些国外的系统。在论坛里，也出现过几次全英文的提问，我想

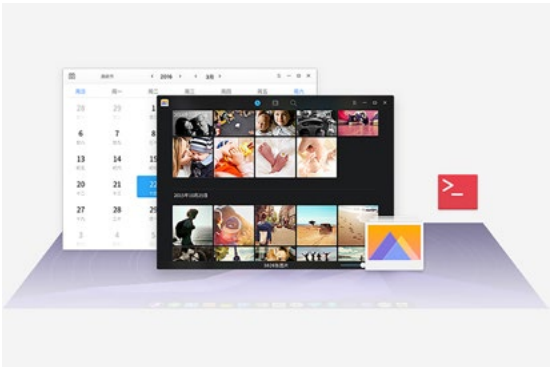




那应该是国外的人通过翻墙进入中国 deepin 论坛提问的吧。

deepin 的论坛，每隔一段时间，关于自主研发浏览器的帖子都会火一下；每隔一段时间，关于浏览器投票的帖子都会重出江湖；每隔一段时间，总会有几个人书写着 deepin 的美。在 deepin 论坛里，你可以看到越来越多的人加入 deepin Linux。会有很多人发着求助的帖子，看到那些帖子，透过那些文字，我好像看到了原来的自己，对着屏幕焦急地抓着头发，迫切的刷新着论坛，认真的看每一层回复，总会有几个人回复：“看楼上。”

如今，我使用 deepin 的时间已经超过了一年，期间系统也崩溃过几次，也想过放弃 Linux，但是，再回到 Windows 的时候，总会习惯性的按 Ctrl+Alt+T，但是没有任何终端出现。于是又重回 Linux 的怀抱，建议新手分区的时候将 /home 单独分区，否则电脑出现问题，重装的话，个人信息都丢失了。现在，我是我们专业电脑技术最好的人，哈哈……我哥都得经常问我，因为我经常敲打代码，我对键盘的熟悉程度也很高，基本上都是盲打的了，除非一些特别生疏的键。



在使用 deepin 的世界里，我真的被改变了好多，如果不用 Linux，我可能会和室友一起开黑，然后拿不到奖学金，会挂科。如果不是因为 deepin，我可能不会学到那么多 Linux 知识。

感谢 deepin，感谢时光让我遇上了你。

期待 15.4 。对了，现在我也能帮助论坛里的人了哦！你如果看到一个蝙蝠侠头像的人在论坛里穿梭，

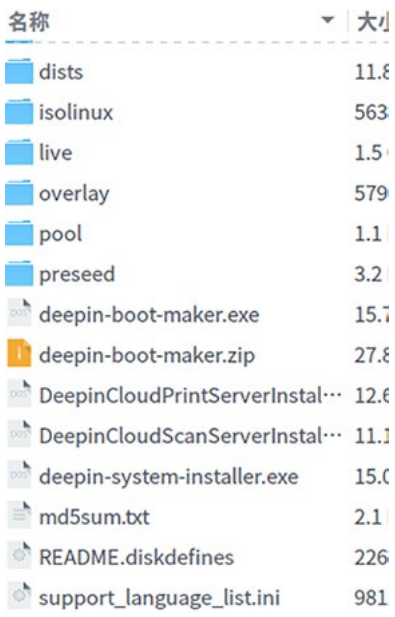
那个人就是我啦。

deepin 15.3 深度操作系统初体验

muqiu / 文

前不久接触了一款基于 linux 开发的国产操作系统 deepin（深度），本着不折腾死不休的心找到了 deepin 官网，初步了解之后，就被这款 os 深深的吸引住了。于是，下载镜像包。奈何手头没有优盘，心累啊！！！

由于体验心切，所以去寻找硬盘装系统的方法，因为本人已经在电脑上装了 kali linux+window7 了。所以百度的方法都不太方便使用。搜索相关资料，各种找方法……正所谓：“车到山前必有路”，正当我准备放弃的时候，点开官方提供的镜像文件看了看。这是什么东西！让我眼前一亮！

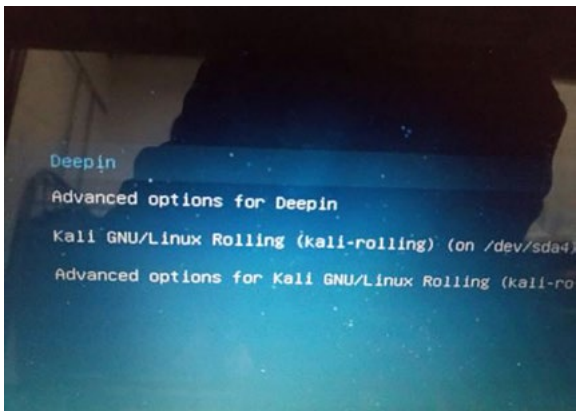


点开看竟然是 exe 文件（deepin-system-installer.exe）。（虽然我英语六级没过，但是这几个单词还是认识的。哈哈哈！）果断点开（本以为需要一个空闲盘，没想到这个安装器居然可以在不清除原有盘符数据的情况下安装，为这么人性化的功能点个赞！）。选好分区，走你！！！

可惜事与愿违，重启没能如愿进入安装界面。分析引导的原因，原来我 kali 用的 grub 引导 window。这个安装过程会更改 MBR 引导。于是，折腾一番，重做 MBR、PBR 引导，再来过。

走你！！！！

如愿进入安装界面。哈哈哈！



不得不说，深度团队很用心，傻瓜似的安装器，不需要用户自己去分各种 /boot ，/swap 等分区。直

接后台自动分配好，非常适合普通用户安装。啥叫人性化！这就叫人性化！

Ps：我现在电脑上有 3 个系统（window7 + kali+deepin），重启开机 deepin 能自动添加 window7 和 kali 的引导（不用再手动修复其他系统的引导了）。来一张 nice 的引导图。

不多说，进入系统。这界面，看着那叫一个 nice 啊！比我之前看到的任何一个 linux 界面都舒服！这 UI 我给 9 分，留下一分怕你骄傲，哈哈！！！

来，先欣赏一下。这是我换了一张壁纸的桌面。

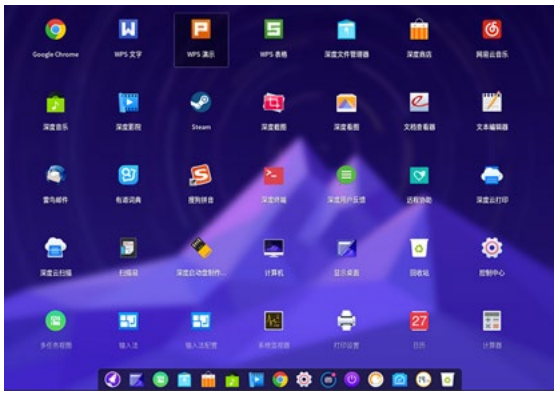
UI 简评：简洁，美观，清新，大气。给人以耳目一新的感觉，让我第一眼就爱上了它。

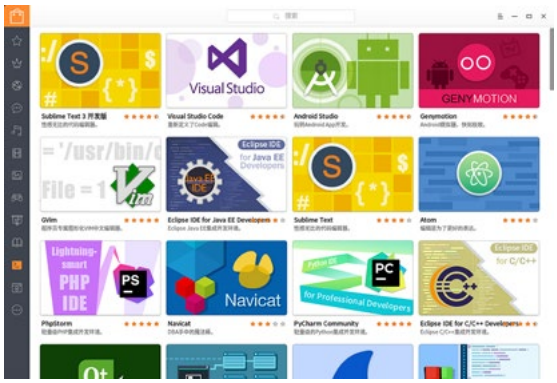
当然，仅仅凭 nice 的 UI 是不足以打动我的。下面进入功能测试。

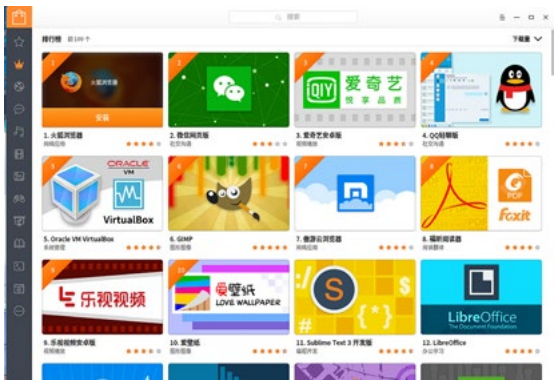
启动器：鼠标移动到最左上角或者点击 super 键就会自动打开菜单（和 kali 一样，人性化），显示方式分两种，一种为分类显示方式，另一种为全平铺模式。加上顶部的搜索框，让使用非常方便。（可是，无法在启动器界面截图……找了一张图。）

粗略一看，QQ、网易云音乐、wps、有道词典、搜狗输入法，这些平时要在 linux 上倒腾很久的东西，现在默认全装好了（人性化！）。换个点开看看，在我这种渣渣配置的老电脑上都没有什么卡顿的痕迹。特别是 QQ，和我在 kali 上的体验完全是两回事！可见 deepin 团队在应用优化适配方面做过很多功夫。

当然，对于程序猿来说，这些功能还不够，如







果没有各种开发语言的 IDE 的话，基本没得玩。于是打开深度应用中心。我天！发现新大陆，这里我只想说一句，老哥稳！！！

首先应用中心做的很美观，分类展示也显得非常有层次。让用户寻找应用更加方便。其次，最重要的是，各种基本软件都有啊，感动得泪流满面啊。

先装一个 sublime 吧，点开，一键安装。（老哥，稳如狗啊！）。再也不用在命令行输入各种难记的命令来安装软件了，而且软件的依赖关系也默认解决好了如此便捷啊，有木有！

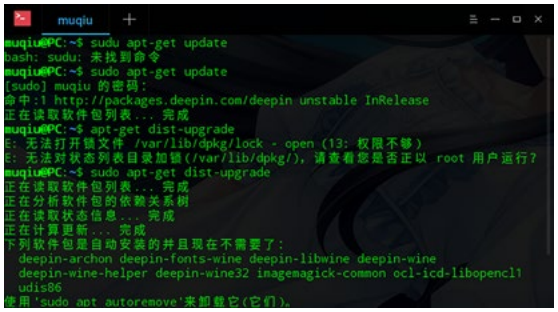
哎，不对，是我眼睛花了么，我咋还看见了安卓版！！！什么黑科技，把安卓的东东搞到这上面跑。吓得我赶紧试了试，发现除了体积大点之外，还真能跑起来！！！（请收下我的膝盖）。

应用部分初步测试到此结束，等我多用几天，再看看有没有其他什么问题。

总的来说，无论是应用的质量还是数量我都给 1 分，剩下 9 分怕你骄傲。啊哈哈哈！不对，我好像说反了（手动滑稽）。讲真，我使用了这么多 linux 发行版桌面操作，包括渗透测试用的 kali、桌面系统做得稍好的 ubuntu、还有 ubuntu 的中国版 kylin ubuntu，唯独 deepin 这款的应用中心才能称得上是正真的应用中心。谁用谁知道。不得不说，deepin 团队在应用开发和适配上做过很多功夫，无论是做基于 wine 的 window 应用兼容，还是开发原生 linux 应用，更有想法的是，创造性的将安卓上的软件移植到 deepin 上。这些工作都为解决 linux 上的应用问题提供了思路。

当然，作为 linux 爱好者，终端使用是必不可少的部分。

ctrl+alt+t，打开终端（半透明界面非常 nice。），测试基本命令都没有问题。



当然，首先是：

Apt-get update

Apt-get dist-upgrade

再说说快捷键，毕竟快捷键的使用能大大提高工作效率：

1、super+ 左右，可以左右切换工作区（个人非常喜欢的功能）。

2、自带截图功能：ctrl+alt+A（反应有点慢，希望下个版本改进下）。

3、文件复制粘贴、窗口切换关闭预览等快捷键都没问题。

4、文件删除快捷键（ctrl+d）不能正常使用……这个，比较尴尬，因为个人比较喜欢使用这个快捷键。

最后简单总结一下吧：

先说缺点：

除了以上提到的小问题之外，还有个 linux 系统的通病：驱动问题。

很多硬件没有对应的驱动支持，导致高配电脑也不能发挥应有的性能。本人几年前的老电脑虽然跑起来还挺流畅，可是看 3D 动漫或者是玩 3D 游戏画面就会明显卡顿（顺便提一句，deepin 下也有不少游戏可以玩的。）。希望 deepin 能与硬件厂商合作，把这个巨坑填上。这样一来，deepin 系统就真的进入‘民用’阶段了。

再说说 deepin 团队：

Deepin 团队真的很用心，能在短短 14 年内做出这么 nice 的系统。已经非常不错了，比某些拿着国家资金美其名曰开发国产操作系统的那帮人强多了。至少，就目前桌面操作来说，deepin 操作系统是我用过的最好用的 linux 操作系统，没有之一。

希望 deepin 团队继续努力，越办越好。将 deepin 发扬光大！

2017 年 2 月 22 日

deepin

I Want You

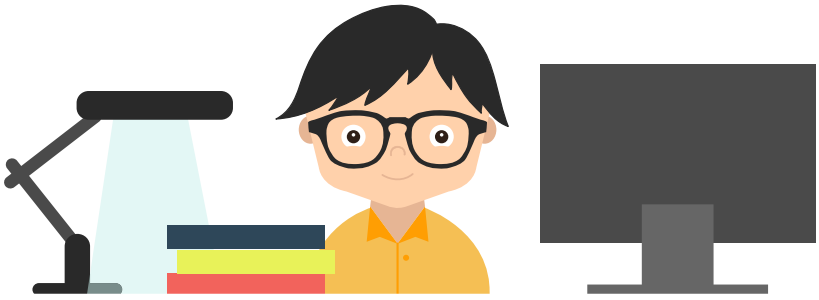
深度书籍社区团队

期待您激情加入

经过一段时间的考虑和规划，为了更好的宣传和了解深度操作系统，同时更好的分享深度操作系统方方面面的知识，我们成立了深度书籍社区团队，该团队主要是为了参与编辑深度书籍、内容大纲（参与定制大纲和讨论）没有太多的限制，希望您的参与！！！

PS：当深度书籍（书名暂定：《给所有人的深度》）编写的程度达到印刷出书要求时，如果发售，我们会将发售收入的一部分用于社区活动奖励，所有参与撰写的用户根据贡献多少给予 1~10 套书籍的赠送。

扫面二维码
填写深度书籍团队申请



deepin 集结 | 征稿启事

《deepin 集结》是深度科技内部刊物，它不仅是记录公司发展历程的一本”画册”，也是公司对外形象和企业建设的窗口。

自 2016 年伊始至今，《deepin 集结》已有定期发放给热爱深度的用户及合作伙伴，并得到了大家的认可。在收集各位读者的反馈意见后，我们开设了新栏目——深度伙伴，主要针对深度员工以外的人员投稿，使《deepin 集结》成为一个企业与用户沟通的刊物，彼此增加交流，分享开源技术。

内部征稿

投稿内容及要求：

形式：摄影作品

Ps. 不要叫你的单反在家睡大觉了，赶紧出来发挥作用吧！

绘画书法

Ps. 你不发出来怎么能知道原来你还这么有才！

诗歌散文、游记、人生感悟

Ps. 知道你原来话不多，但是肚子里面还是有墨水的！

经验技术分享

Ps. 专业达人们，把你们的专业知识拿来 show 一下吧，科普一下啦！

以及能落实到纸上的任何才艺

Ps. 还有多少是我想不到的，快点告诉我！

外部征稿

投稿内容及要求：

1. 内容：a 技术分享
   b 用户体验
   c 项目评价
2. 要求：a 与行业相关
   b 字数 1000–4000
   c 文中图片需清晰

深度的同学可以告知身边爱好开源的发烧友积极投稿哦！

投稿邮件主题需标明： 外部投稿（字样）+ 姓名 + 手机号码

要求：我们很看重有图有真相哦，图片和文字说明一样重要。

稿酬：现金（微信红包）或精美礼品

投稿方式：deepin–magazine@deepin.com 邮件名称一定要注明 “所在城市 – 部门 – 姓名”

诸位同学有任何问题，都可以立即马上咨询我们：

qindi@deepin.com

deepin

we do we change...