# GNU linux 趣话

李智华

中国科学院 开源协会

October 29, 2011

# 大纲

# 中科院开源协会宗旨

1. 自由软件精神：自由、开放、共享——Chase freedom and share your wisdom
2. ubuntu 精神：人道待人——天下共享连接人人的信念

具有 ubuntu 精神的人心胸开阔，乐于助人，见贤思齐而不忌妒贤能，因为他或她拥有适度的自信，而这源自如下认识：自己属于一个更大的整体，当他人受到伤害或死去时，当他人受到折磨或压迫时，这个整体就会消失。”——大主教 Desmond Tutu

# 中科院开源协会宗旨

1. 自由软件精神：自由、开放、共享——Chase freedom and share your wisdom
2. ubuntu 精神：人道待人——天下共享连接人人的信念

具有 ubuntu 精神的人心胸开阔，乐于助人，见贤思齐而不忌妒贤能，因为他或她拥有适度的自信，而这源自如下认识：自己属于一个更大的整体，当他人受到伤害或死去时，当他人受到折磨或压迫时，这个整体就会消失。”——大主教 Desmond Tutu

返回

# 什么是 Linux(1/3)

在如今的操作系统市场上，除了一些专业的领域以外，可以说微软正在以其 Windows 操作系统的强劲攻势横扫全球市场，能与其相抗衡的公司微乎其微。

但是在迅猛发展的国际互联网上，有这样一群人，他们是一支由编程高手，业余计算机玩家，黑客们组成的奇怪队伍，完全独立地开发出在功能上毫不逊色于微软的商业操作系统的一个全新的免费 UNIX 操作系统——Linux（发音为 Li-nucks)[1]。

[1]参考：http://hepg.sdu.edu.cn/Service/linux/intro/intro.html

# linux 是什么 (2/3)

Wiki 解释：Linux(也被称为 GNU/Linux) 是一种自由和开放源代码的计算机操作系统。

Linux 是一个领先的操作系统，世界上运算最快的 10 台超级计算机运行的都是 Linux 操作系统[2]。

Linux 操作系统也是自由软件和开放源代码发展中最著名的例子。 只要遵循 GNU 通用公共许可证, 任何人和机构都可以自由地使用 Linux 的所有底层源代码，也可以自由地修改和再发布。

[2]参考：http://zh.wikipedia.org/zh-cn/Linux

# linux 是什么 (3/3)

LAMP 是一个缩写，它指一组通常一起使用来运行动态网站或者服务器的自由软件：

1. Linux，操作系统；
2. Apache，网页服务器；
3. MySQL，数据库管理系统（或者数据库服务器）；
4. PHP 和有時 Perl 或 Python，脚本语言。

举例来说，Wikipedia，免费自由的百科全书，运行的一系列软件具有 LAMP 环境一样的特点。Wikipedia 使用 MediaWiki 软件，主要在 Linux 下开发，由 Apache HTTP 服务器提供内容，在 MySQL 数据库中存储内容，PHP 来实现程序逻辑[3]。

[3]参考：http://zh.wikipedia.org/wiki/LAMP

# linux 历史[4]

为什么大家常常会说，Linux 是很稳定的一套操作系统呢？这是因为，Linux 他有个老前辈，那就是 Unix 家族。我们来谈一谈 Unix 到 Linux 的这一段历史吧。

早在 Linux 出现之前的二十年（大约在 1970 年代)，就有一个相当稳定而成熟的操作系统存在了！那就是 Linux 的老大哥『Unix 』是也！怎么这么说呢？！他们这两个家伙有什么关系呀？这里就给他说一说啰！众所皆知的，Linux 的核心是由 Linus Torvalds 在 1991 年的时候给他开发出来的，并且丢到网络上提供大家下载，后来大家觉得这个小东西（ Linux Kernel ）相当的小而精巧，所以慢慢的就有相当多的朋友投入这个小东西的研究领域里面去了！但是为什么这的小东西这么棒呢？！然而又为什么大家都可以免费的下载这个东西呢？！

[4]参考《鸟哥的 linux 私房菜》http://linux.vbird.org/linux_basic/0110whatislinux.php#whatislinux_os

# 1969 年以前，一个伟大的梦想：Bell,MIT 与 GE 的『Multics』系统 (1/2)

早期的电脑并不像现在的个人电脑一样普遍。非但如此，早期的电脑架构还很难使用，除了运算速度并不快之外，操作介面也很困扰的！因为那个时候的输入设备只有读卡机、输出设备只有印表机，使用者也无法与作业系统互动。在那之后，由于硬件与作业系统的改良，使得后来可以使用键盘来进行输入。不过，在一间学校里面，主机毕竟可能只有一部，如果多人等待使用，那怎么办？大家还是得要等待啊！好在 1960 年代初期麻省理工学院 (MIT) 发展了所谓的：『相容分时系统(Compatible Time-Sharing System, CTSS)』，它可以让大型主机透过提供数个终端机 (terminal) 以连线进入主机，来利用主机的资源进行运算工作。架构有点像这样：

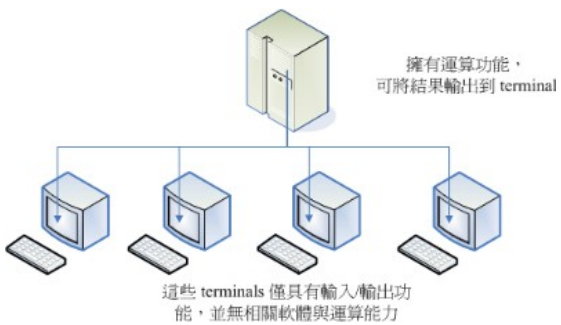


Figure: 早期主机与终端机的相关性图示

如此一来，无论主机在哪里，只要在终端机前面进行输入输出的作业，就可利用主机提供的功能了。不过，需要注意的是，此时终端机只具有输入/输出的功能，本身完全不具任何运算或者软件安装的能力。而且，比较先进的主机大概也只能提供 30 个不到的终端机而已。

# 1969 年以前，一个伟大的梦想：Bell,MIT 与 GE 的『Multics』系统 (2/2)

为了更加强化大型主机的功能，以让主机的资源可以提供更多使用者来利用，所以在 1965 年前后，由贝尔实验室 (Bell)、麻省理工学院 (MIT) 及奇异公司 (GE, 或称为通用电器) 共同发起了 Multics[5]的计划，Multics 计划的目的是想要让大型主机可以达成提供 300 个以上的终端机连线使用的目标。不过，到了 1969 年前后，计划进度落后，资金也短缺，所以该计划虽然继续在研究，但贝尔实验室还是退出了该计划的研究工作[6]。

[5] Multics 有复杂、多数的意思存在

[6] Tips: 最终 Multics 还是有成功的发展出他们的系统，完整的历史说明可以参考：http://www.multicians.org/网站内容。Multics 计划虽然后来没有受到很大的重视，但是他培养出来的人才是相当优秀的！

# 1969 年：Ken Thompson 的小型 file server system(1/2))

在认为 Multics 计划不可能成功之后，贝尔研究室就退出该计划。不过，原本参与 Multics 计划的人员中，已经从该计划当中获得一些点子，Ken Thompson 就是其中一位！

Thompson 因为自己的需要，希望开发一个小小的作业系统以提供自己的需求。在开发时，有一部 DEC(Digital Equipment Corporation) 公司推出的 PDP-7 刚好没人使用，于是他就准备针对这部主机进行作业系统核心程序的撰写。本来 Thompson 应该是没时间的 (有家有小孩的宿命？)，无巧不巧的是，在 1969 年八月份左右，刚好 Thompson 的妻儿去了美西探亲，于是他有了额外的一个月的时间好好的待在家将一些构想实现出来！

# 1969 年：Ken Thompson 的小型 file server system(2/2)

经过四个星期的奋斗，他终于以组合语言 (Assembler) 写出了一组核心程序，同时包括一些核心工具程序，以及一个小小的档案系统。那个系统就是 Unix 的原型！当时 Thompson 将 Multics 庞大的复杂系统简化了不少，于是同实验室的朋友都戏称这个系统为：Unics。(当时尚未有 Unix 的名称)

Thompson 的这个档案系统有两个重要的概念，分别是：

1. 所有的程序或系统装置都是档案
2. 不管建构编辑器还是附属档案，所写的程序只有一个目的，且要有效的完成目标。

这些概念在后来对于 Linux 的发展有相当重要的影响！

Tips: 当初 Thompson 会写这套 Unix 核心程序，却是想要移植一套名为『太空旅游』的游戏呢！

# 1973 年：Unix 的正式诞生，Ritchie 等人以 C 语言写出第一个正式 Unix 核心

由于 Thompson 写的那个作业系统实在太好用了，所以在贝尔实验室内部广为流传，并且数度经过改版。但是因为 Unics 本来是以组合语言写成的，组合语言具有专一性，加上当时的电脑机器架构都不太相同，所以每次要安装到不同的机器都得要重新编写组合语言，真不方便！

**后来 Thompson 与 Ritchie 合作想将 Unics 改以高级程序语言来撰写。当时现成的高级程序语言有 B 语言。但是由 B 语言所编译出来的核心效能不是很好。后来 Dennis Ritchie 将 B 语言重新改写成 C 语言，再以 C 语言重新改写与编译 Unics 的核心，最后正名与发行出 Unix 的正式版本！**

需要特别强调的是，由于 Unix 是以较高级的 C 语言写的，相对于组合语言需要与硬件有密切的配合，高级的 C 语言与硬件的相关性就没有这么大了！所以，这个改变也使得 Unix 很容易被移植到不同的机器上面！

Tips: 这群高级骇客实在很厉害！因为自己的需求来开发出这么多好用的工具！C 程序语言开发成功后，甚至一直沿用至今！你说厉不厉害啊！这个故事也告诉我们，不要小看自己的潜能！你想作的，但是现实生活中没有的，就动手自己搞一个来玩玩吧！

# Dennis Ritchie——C 语言之父,UNIX 之父[7](1/3))

丹尼斯·麦卡利斯泰尔·里奇（Dennis MacAlistair Ritchie，1941 年 9 月 9 日 – 2011 年 10 月 12 日），出生于美国纽约。著名的美国计算机科学家，对 C 语言和其他编程语言、Multics 和 Unix 等操作系统的发展做出了巨大贡献。里奇在哈佛大学学习物理学和应用数学毕业，1967 年他进入贝尔实验室，他曾是朗讯技术公司系统软件研究部门的领导人（2007 年退休）。1983 年他与肯·汤普逊一起获得了图灵奖。理由是他们"研究发展了通用的操作系统理论，尤其是实现了 UNIX 操作系统"。1999 年两人为发展 C 语言和 Unix 操作系统一起获得了美国国家技术奖章。

Figure: Dennis Ritchie

[7]参考：http://zh.wikipedia.org/wiki/丹尼斯·里奇

# Dennis Ritchie——C 语言之父,UNIX 之父[8](2/3))

## Dennis Ritchie 生平：

1. 1941 年 9 月 9 日，丹尼斯 · 里奇出生。曾在哈佛大学学习物理学和应用数学。
2. 1967 年，里奇进入贝尔实验室，生前曾经担任朗讯技术公司系统软件研究部门的领导人。
3. 1978 年，里奇与布莱恩 · 科尔尼干（Brian W. Kernighan）一起出版了名著《C 程序设计语言（The C Programming Language)》，现在此书已翻译成多种语言，成为 C 语言方面最权威的教材之一。这本书被程序员们称为“白皮书”，获得狂热拥戴。然而，由于 C 语言的简洁和高效，也成为入侵他人电脑的利器之一。里奇因此被诸多电脑黑客尊为导师，虽然里奇本人并不认可这一说法。
4. 1983 年，美国计算机协会将当年的图灵奖破例颁给了作为软件工程师的肯 · 汤普逊与里奇，获奖原因是他们“研究发展了通用的操作系统理论，尤其是实现了 Unix 操作系统”。并且，美国计算机协会当年还决定新设立一个奖项——软件系统奖，以奖励那些优秀的软件开发者。
5. 2011 年 10 月 9 日，丹尼斯 · 里奇去世，享年 70 岁。
6. 2011 年 10 月 13 日，在众多的国际互动论坛上，计算机爱好者们以特有的方式纪念这位编程语言的重要奠基人。许多网友的发帖中没有片言只字，仅仅留下一个分号“ ;”。在 C 语言中，分号标志着一行指令语句的结束，网友们以此来悼念“C 语言之父”，美国著名计算机专家丹尼斯 · 里奇（Dennis Ritchie）所引领的时代远去。

[8]参考：http://www.hudong.com/wiki/丹尼斯 · 里奇

# Dennis Ritchie——C 语言之父,UNIX 之父[9](3/3))

乔布斯去世时，网络上铺天盖地诸多赞誉与哀思，其产品风靡所带来的全球性用户崇拜史无前例。其实，里奇先生更应享受这些赞誉，甚至更多。

缺少了里奇所创造的 C 语言和 UNIX ，网络和任何网络产品都不可能存在。比如，浏览器是用 C 语言写的，网络服务器是 C 语言写的，很多人反驳说他们所使用的是 JAVA 或者 C++，但它们也是 C 语言的衍生物。包括网页架构时的 Python 和 Ruby 两种程序语言，也是基于 C 语言的。除此而外，所有的网络硬件产品都是 C 语言所编写的，而 UNIX 是整个因特网所运行的基础。由此可见，我们怎样评价里奇先生的丰功伟绩，都不足为过。甚至 Windows 也曾经用 C 语言写过，苹果公司用于个人电脑、iPad 和 iPhone 上的操作系统，都是基于 UNIX 的。

麻省理工大学计算机系的马丁教授评价说：" 如果说，乔布斯是可视化产品中的国王，那么里奇就是不可见王国中的君主。乔布斯的贡献在于，他如此了解用户的需求和渴求，以至于创造出了让当代人乐不思蜀的科技产品。然而，却是里奇先生为这些产品提供了最核心的部件，人们看不到这些部件，却每天都在使用着。"

[9]参考：http://www.guokr.com/article/67488/

# 1977 年：重要的 Unix 分支——BSD 的诞生

虽然贝尔属于 AT & T(美国电信大厂)，但是 AT & T 此时对于 Unix 是采取较开放的态度，此外，Unix 是以高级的 C 语言写成的，理论上是具有可移植性的！亦即只要取得 Unix 的原始码，并且针对大型主机的特性加以修订原有的原始码 (Source Code)，就可能将 Unix 移植到另一部不同的主机上头了。所以在 1973 年以后，Unix 便得以与学术界合作开发！最重要的接触就是与加州柏克莱 (Berkeley) 大学的合作了。

柏克莱大学的 Bill Joy 在取得了 Unix 的核心原始码后，着手修改成适合自己机器的版本，并且同时增加了很多任务具软件与编译程序，最终将它命名为 Berkeley Software Distribution (BSD)。这个 BSD 是 Unix 很重要的一个分支，Bill Joy 也是 Unix 业者『Sun(升阳)』这家公司的创办者！Sun 公司即是以 BSD 发展的核心进行自己的商业 Unix 版本的发展的。(后来可以安装在 x86 硬件架构上面 FreeBSD 即是 BSD 改版而来！)

# 1979 年：重要的 System V 架构与版权宣告 (1/2)

由于 Unix 的高度可移植性与强大的效能，加上当时并没有版权的纠纷，所以让很多商业公司开始了 Unix 作业系统的发展，例如 AT & T 自家的 System V、IBM 的 AIX 以及 HP 与 DEC 等公司，都有推出自家的主机搭配自己的 Unix 作业系统[10]。

另外，由于没有厂商针对个人电脑设计 Unix 系统，因此，在早期□□并没有支援个人电脑的 Unix 作业系统的出现。

Tips: 如同相容分时系统的功能一般，Unix 强调的是多人多任务的环境！但早期的 286 个人电脑架构下的 CPU 是没有能力达到多任务的作业，因此，并没有人对移植 Unix 到 x86 的电脑上有兴趣。

[10] 在早期每一家生产电脑硬件的公司还没有所谓的『协议』的概念，所以每一个电脑公司出产的硬件自然就不相同啰！因此他们必须要为自己的电脑硬件开发合适的 Unix 系统。例如在学术机构相当有名的 Sun、Cray 与 HP 就是这一种情况。他们开发出来的 Unix 作业系统以及内含的相关软件并没有办法在其他的硬件架构下工作的！

# 1979 年：重要的 System V 架构与版权宣告 (2/2)

每一家公司自己出的 Unix 虽然在架构上面大同小异，但是却真的仅能支援自身的硬件，所以啰，早先的 Unix 只能与伺服器 (Server) 或者是大型工作站 (Workstation) 划上等号！但到了 1979 年时，AT & T 推出 System V 第七版 Unix 后，这个情况就有点改善了。这一版最重要的特色是可以支援 x86 架构的个人电脑系统，也就是说 System V 可以在个人电脑上面安装与运作了。

不过因为 AT & T 由于商业的考量，以及在当时现实环境下的思考，于是想将 Unix 的版权收回去。因此，AT & T 在 1979 年发行的第七版 Unix 中，特别提到了『不可对学生提供原始码』的严格限制！同时，也造成 Unix 业界之间的紧张气氛，并且也引爆了很多的商业纠纷。

Tips: 目前被称为纯正的 Unix 指的就是 System V 以及 BSD 这两套！

# 1984 年之一：x86 架构的 Minix 作业系统诞生 (1/2)

关于 1979 年的版权声明中，影响最大的当然就是学校教 Unix 核心原始码相关学问的教授了！想一想，如果没有核心原始码，那么如何教导学生认识 Unix 呢？这问题对于 Andrew Tanenbaum 教授来说，实在是很伤脑筋的！不过，学校的课程还是得继续啊！那怎么办？

既然 1979 年的 Unix 第七版可以在 Intel 的 x86 架构上面进行移植，那么是否意味着可以将 Unix 改写并移植到 x86 上面了呢？在这个想法上，Tanenbaum 教授于是乎自己动手写了 Minix 这个 Unix Like 的核心程序！在撰写的过程中，为了避免版权纠纷，Tanenbaum 完全不看 Unix 核心原始码！并且强调他的 Minix 必须能够与 Unix 相容才行！Tanenbaum 在 1984 年开始撰写核心程序，到了 1986 年终于完成，并于次年出版 Minix 相关书籍。

Tips: 之所以称为 Minix 的原因，是因为他是个 Mini 的 Unix 系统！

# 1984 年之一：x86 架构的 Minix 作业系统诞生 (2/2)

这个 Minix 版本比较有趣的地方是，他并不是完全免费的，无法在网路上提供下载！必须要透过磁片/磁带购买才行！虽然真的很便宜。

不过，毕竟因为没有在网路上流传，所以 Minix 的传递速度并没有很快速！此外，购买时，随磁片还会附上 Minix 的原始码！这意味着使用者可以学习 Minix 的核心程序设计概念！(这个特色对于 Linux 的启始开发阶段，可是有很大的关系！)

此外，Minix 作业系统的开发者仅有 Tanenbaum 教授，因为学者很忙啊！加上 Tanenbaum 始终认为 Minix 主要用在教育用途上面，所以对于 Minix 是点到为止！

# 1984 年之二：GNU 计划与 FSF 基金会的成立 (1/5)

Richard Mathew Stallman(史托曼) 在 1984 年发起的 GNU 计划，对于现今的自由软件风潮，真有不可磨灭的地位！目前我们所使用得很多自由软件，几乎均直接或间接受益于 GNU 这个计划！那么史托曼是何许人也？为何他会发起这个 GNU 计划呢？

Figure: Richard Mathew Stallman

# 1984 年之二：GNU 计划与 FSF 基金会的成立 (2/5)

一个分享的环境：

1. Richard Mathew Stallman(生于 1953 年，网路上自称的 ID 为 RMS) 从小就很聪明！他在 1971 年的时候，进入骇客圈中相当出名的人工智慧实验室 (AI Lab.)，这个时候的骇客专指电脑功力很强的人，而非破坏电脑的怪客 (cracker)！
2. 当时的骇客圈对于软件的著眼点几乎都是在『分享』，所以并没有专利方面的困扰！这个特色对于史托曼的影响很大！不过，后来由于管理阶层的问题，导致实验室的优秀骇客离开该实验室，并且进入其他商业公司继续发展优秀的软件。但史托曼并不服输，仍然持续在原来的实验室开发新的程序与软件。后来，他发现到，自己一个人并无法完成所有的工作，于是想要成立一个开放的团体来共同努力！

# 1984 年之二：GNU 计划与 FSF 基金会的成立 (3/5)

## 使用 Unix 开发阶段：

1. 1983 年以后，因为实验室硬件的更换，使得史托曼无法继续以原有的硬件与作业系统继续自由程序的撰写～而且他进一步发现到，过去他所使用的 Lisp 作业系统，是麻省理工学院的专利软件，是无法共享的，这对于想要成立一个开放团体的史托曼是个阻碍。于是他便放弃了 Lisp 这个系统。后来，他接触到 Unix 这个系统，并且发现，Unix 在理论与实际上，都可以在不同的机器间进行移植。虽然 Unix 依旧是专利软件，但至少 Unix 架构上还是比较开放的！于是他开始转而使用 Unix 系统。
2. 因为 Lisp 与 Unix 是不同的系统，所以，他原本已经撰写完毕的软件是无法在 Unix 上面运行的！为此，他就开始将软件移植到 Unix 上面。并且，为了让软件可以在不同的平台上运作，因此，史托曼将他发展的软件均撰写成可以移植的型态！也就是他都会将程序的原始码公布出来！

# 1984 年之二：GNU 计划与 FSF 基金会的成立 (4/5)

GNU 计划的推展：

1. 1984 年，史托曼开始 GNU 计划，这个计划的目的是：建立一个自由、开放的 Unix 作业系统 (Free Unix)。但是建立一个作业系统谈何容易啊！而且在当时的 GNU 是仅有自己一个人单打独斗的史托曼～这实在太麻烦，但又不想放弃这个计划，那可怎么办啊？
2. 聪明的史托曼干脆反其道而行～『既然作业系统太复杂，我就先写可以在 Unix 上面运行的小程序，这总可以了吧？』在这个想法上，史托曼开始参考 Unix 上面现有的软件，并依据这些软件的作用开发出功能相同的软件，且开发期间史托曼绝不看其他软件的原始码，以避免吃上官司。后来一堆人知道免费的 GNU 软件，并且实际使用后发现与原有的专利软件也差不了太多，于是便转而使用 GNU 软件，于是 GNU 计划逐渐打开知名度。

# 1984 年之二：GNU 计划与 FSF 基金会的成立 (4/5)

## GNU 计划的推展（续）：

1. 虽然 GNU 计划渐渐打开知名度，但是能见度还是不够。这时史托曼又想：不论是什么软件，都得要进行编译成为二进位档案 (binary program) 后才能够执行，如果能够写出一个不错的编译器，那不就是大家都需要的软件了吗？因此他便开始撰写 C 语言的编译器，那就是现在相当有名的GNU C Compiler(gcc)！这个点相当的重要！这是因为 C 语言编译器版本众多，但都是专利软件，如果他写的 C 编译器够棒，效能够佳，那么将会大大的让 GNU 计划出现在众人眼前！
2. 但开始撰写 GCC 时并不顺利，为此，他先转而将他原先就已经写过的Emacs 编辑器写成可以在 Unix 上面跑的软件，并公布原始码。Emacs 是一种程序编辑器，他可以在使用者撰写程序的过程中就进行程序语法的检验，此一功能可以减少程序设计师除错的时间！因为 Emacs 太优秀了，因此，很多人便直接向他购买[11]。
3. 此时网际网路尚未流行，所以，史托曼便借着 Emacs 以磁带 (tape) 出售，赚了一点钱，进而开始全力撰写其他软件。并且成立自由软件基金会 (FSF, Free Software Foundation)，请更多任务程师与志工撰写软件。终于还是完成了 GCC，这比 Emacs 还更有帮助！此外，他还撰写了更多可以被调用的 C 函式库 (GNU C library)，以及可以被使用来操作作业系统的基本介面BASH shell！这些都在 1990 年左右完成了！

[11] Tips: 如果纯粹使用文字编辑器来编辑程序的话，那么程序语法如果写错时，只能利用编译时发生的错误讯息来修订了，这样实在很没有效率。Emacs 则是一个很棒的编辑器！他可以很快的立刻显示出你写入的语法可能有错误的地方，这对于程序设计师来说，实在是一个好到不能再好的工具了！所以才会这么的受到欢迎啊！

# 1984 年之二：GNU 计划与 FSF 基金会的成立 (5/5)

到了 1985 年，为了避免 GNU 所开发的自由软件被其他人所利用而成为专利软件，所以他与律师草拟了有名的通用公共许可证 (General Public License, GPL)，并且称呼他为 copyleft(相对于专利软件的 copyright！)。关于 GPL 的相关内容我们在后面继续谈论，在这里，必须要说明的是，由于有 GNU 所开发的几个重要软件，如：Emacs、GNU C (GCC)、GNU C Library (glibc)、Bash shell, 造成后来很多的软件开发者可以藉由这些基础的工具来进行程序开发！进一步壮大了自由软件团体！这是很重要的！

不过，对于 GNU 的最初构想『建立一个自由的 Unix 作业系统』来说，有这些优秀的程序是仍无法满足，因为，当下并没有『自由的 Unix 核心』存在... 所以这些软件仍只能在那些有专利的 Unix 平台上工作, 一直到 Linux 的出现...

# 1988 年：图形介面 XFree86 计划

有鉴于图形使用者介面 (Graphical User Interface, GUI) 的需求日益加重，在 1984 年由 MIT 与其他协力厂商首次发表了 X Window System ，并且更在 1988 年成立了非营利性质的 XFree86 这个组织。所谓的 XFree86 其实是 X Window System + Free + x86 的整合名称呢！而这个 XFree86 的 GUI 介面更在 Linux 的核心 1.0 版于 1994 年释出时，整合于 Linux 作业系统当中！ Tips: 为什么称图形使用者介面为 X 呢？因为由英文单字来看，Window 的 W 接的就是 X 啦！意指 Window 的下一版就是了！需注意的是，X Window 并不是 X Windows！

# 1991 年：芬兰大学生 Linus Torvalds 的一则简讯

到了 1991 年，芬兰的赫尔辛基大学的 Linus Torvalds 在 BBS 上面贴了一则消息，宣称他以 bash, gcc 等工具写了一个小小的核心程序，这个核心程序可以在 Intel 的 386 机器上面运作，让很多人很感兴趣！从此开始了 Linux 不平凡的路程！

# GNU 与 GPL(1/5)——自由软件的活动

GNU 计划对于整个自由软件来说是占有非常重要的角色！底下我们就来谈谈吧！

1984 年创立 GNU 计划与 FSF 基金会的 Stallman 先生认为，写程序最大的快乐就是让自己发展的良好的软件让大家来使用了！而既然程序是想要分享给大家使用的，不过，每个人所使用的电脑软硬件并不相同，既然如此的话，那么该程序的原始码 (Source code) 就应该要同时释出，这样才能方便大家修改而适用于每个人的电脑中呢！这个将原始码连同软件程序释出的举动，就称为自由软件 (Free Software) 运动！

此外，史托曼同时认为，如果你将你程序的 Source code 分享出来时，若该程序是很优秀的，那么将会有很多人使用，而每个人对于该程序都可以查阅 source code，无形之中，就会有一票人帮助你除错啰！你的这支程序将会越来越壮大！越来越优秀呢！

# GNU 与 GPL(2/5)——自由软件的版权 GNU GPL

为了避免自己的开发出来的 Open source 自由软件被拿去做成专利软件，于是 Stallman 同时将 GNU 与 FSF 发展出来的软件，都挂上 GPL 的版权宣告～这个 FSF 的核心观念是『版权制度是促进社会进步的手段，版权本身不是自然权力。』

Tips: 为什么要称为 GNU 呢？其实 GNU 是 GNU's Not Unix 的缩写，意思是说，GNU 并不是 Unix 啊！那么 GNU 又是什么呢？就是 GNU's Not Unix 嘛！..... 如果你写过程序就会知道，这个 GNU = GNU's Not Unix 可是无穷递归啊！

另外，什么是 Open Source 呢？所谓的 source 是程序发展者写出的原始程序码，Open Source 就是，软件在发布时，同时将作者的原始码一起公布的意思！

# GNU 与 GPL(3/5)——自由 (Free) 的真谛

那么这个 GPL(GNU General Public License, GPL) 是什么？为什么要将自由软件挂上 GPL 的『版权宣告』呢？这个版权宣告对于作者有何好处？首先，Stallman 对 GPL 一直是强调 Free 的，这个 Free 的意思是这样的："Free software" is a matter of liberty, not price. To understand the concept, you should think of "free speech", not "free beer". "Free software" refers to the users' freedom to run, copy, distribute, study, change, and improve the software.(大意是说，Free Software(自由软件) 是一种自由的权力，并非是『价格！』举例来说，你可以拥有自由呼吸的权力、你拥有自由发表言论的权力，但是，这并不代表你可以到处喝『免费的啤酒！(free beer)』，也就是说，自由软件的重点并不是指『免费』的，而是指具有『自由度, freedom』的软件，史托曼进一步说明了自由度的意义是：使用者可以自由的执行、复制、再发行、学习、修改与强化自由软件。这无疑是个好消息！因为如此一来，你所拿到的软件可能原先只能在 Unix 上面跑，但是经过原始码的修改之后，你将可以拿他在 Linux 或者是 Windows 上面来跑！)

# GNU 与 GPL(3/5)——自由 (Free) 的真谛

总之，一个软件挂上了 GPL 版权宣告之后，他自然就成了自由软件！这个软件就具有底下的特色：

1. 取得软件与原始码：你可以根据自己的需求来执行这个自由软件；
2. 复制：你可以自由的复制该软件；
3. 修改：你可以将取得的原始码进行程序修改工作，使之适合你的工作；
4. 再发行：你可以将你修改过的程序，再度的自由发行，而不会与原先的撰写者冲突；
5. 回馈：你应该将你修改过的程序码回馈于社群！

但请特别留意，你所修改的任何一个自由软件都不应该也不能这样：

1. 修改授权：你不能将一个 GPL 授权的自由软件，在你修改后而将他取消 GPL 授权～
2. 单纯贩卖：你不能单纯的贩卖自由软件。

# GNU 与 GPL(3/5)——自由 (Free) 的真谛

也就是说，既然 GPL 是站在互助互利的角度上去开发的，你自然不应该将大家的成果占为己有，对吧！因此你当然不可以将一个 GPL 软件的授权取消，即使你已经对该软件进行大幅度的修改！

那么自由软件也不能贩卖吗？当然不是！GPL 是可以从事商业行为的。还记得上一个小节里面，我们提到史托曼藉由贩卖 Emacs 取得一些经费，让自己生活不至于匮乏吧？是的！自由软件是可以贩售的，不过，不可仅贩售该软件，应同时搭配售后服务与相关手册。

# GNU 与 GPL(4/5)——自由 (Free) 的好处

1. 既然是 Open Source 的自由软件，那么你的程序码将会有很多人帮□□你查阅，如此一来，程序的漏洞与程序的优化将会进展的很快！所以，在安全性与效能上面，自由软件一点都不输给商业软件！此外，因为 GPL 授权当中，修改者并不能修改授权，因此，你如果曾经贡献过程序码，嘿嘿！你将名留青史呢！不错吧！
2. 对于程序开发者来说，GPL 实在是一个非常好的授权，因为大家可以互相学习对方的程序撰写技巧，而且自己写的程序也有人可以帮忙除错。
3. 那你会问啊，对于广大的终端用户，GPL 有没有什么好处啊？有啊！当然有！虽然终端用户或许不会自己编译程序码或者是帮人家除错，但是终端用户使用的软件□□绝大部分就是 GPL 的软件，全世界有一大票的工程师在帮助你维护你的系统，这难道不是一件非常棒的事吗？

# GNU 与 GPL(5/5)——Richard Matthew Stallman[12]

理查·马修·斯托曼（Richard Matthew Stallman，简称 rms，1953 年 3 月 16 日 –）是美国自由软件运动的精神领袖、GNU 计划以及自由软件基金会（Free Software Foundation）的创立者。作为一个著名的黑客，他的主要成就包括 Emacs 及后来的 GNU Emacs，GNU C 编译器及 GDB 调试器。他所写作的 GNU 通用公共许可证（GNU GPL）是世上最广为采用的自由软件许可证，为 copyleft 观念开拓出一条崭新的道路。

他最大的影响是为自由软件运动竖立道德、政治及法律框架。他被许多人誉为当今自由软件的斗士、伟大的理想主义者。

[12]参考：http://zh.wikipedia.org/wiki/理查德·斯托曼

# Torvalds 的 Linux 发展 (1/3)

我们前面一节当中，提到了 Unix 的历史，也提到了 Linux 是由 Torvalds 这个芬兰人所发明的。那么为何托瓦兹可以发明 Linux 呢？凭空想像而来的？还是有什么渊源？这里我们就来谈一谈！


Figure: Dennis Ritchie

# Torvalds 的 Linux 发展 (1/3)——与 Minix 之间

Linus Torvalds(托瓦兹, 1969 年出生) 的外祖父是赫尔辛基大学的统计学家，他的外祖父为了让自己的小孙子能够学点东西，所以从小就将托瓦兹带到身边来管理一些微电脑。在这个时期，托瓦兹接触了组合语言 (Assembly Language)，那是一种直接与晶片对谈的程序语言，也就是所谓的低阶语言。必须要很了解硬件的架构，否则很难以组合语言撰写程序的。

在 1988 年间，托瓦兹顺利的进入了赫尔辛基大学，并选读了电脑科学系。在就学期间，因为学业的需要与自己的兴趣，托瓦兹接触到了 Unix 这个作业系统。当时整个赫尔辛基只有一部最新的 Unix 系统，同时仅提供 16 个终端机 (terminal)。还记得我们上一节刚刚提过的，早期的电脑仅有主机具有运算功能，terminal 仅负责提供 Input/Output 而已。在这种情况下，实在很难满足托瓦兹的需求，因为..... 光是等待使用 Unix 的时间，就很耗时～为此，他不禁想到：『我何不自己搞一部 Unix 来玩？』不过，就如同 Stallman 当初的 GNU 计划一样，要写核心程序，谈何容易～

不过，幸运之神并未背离托瓦兹，因为不久之后，他就知道有一个类似 Unix 的系统，并且与 Unix 完全相容，还可以在 Intel 386 机器上面跑的作业系统，那就是我们上一节提过的，谭宁邦教授为了教育需要而撰写的 Minix 系统！他在购买了最新的 Intel 386 的个人电脑后，就立即安装了 Minix 这个作业系统。另外，上个小节当中也谈到，Minix 这个作业系统是有附上原始码的，所以托瓦兹也经由这个原始码学习到了很多的核心程序设计的设计概念喔！

# Torvalds 的 Linux 发展 (2/3)——对 386 硬件的多任务测试

事实上，托瓦兹对于个人电脑的 CPU 其实并不满意，因为他之前碰的电脑都是工作站型的电脑，这类电脑的 CPU 特色就是可以进行『多任务处理』的能力。什么是多任务呢？理论上，一个 CPU 在一个时间内仅能进行一个程序，那如果有两个以上的程序同时出现到系统中呢？举例来说，你可以在现今的电脑中同时开启两个以上的办公软件，例如电子试算表与文书处理软件。这个同时开启的动作代表着这两□□个程序同时要交给 CPU 来处理～

早期 Intel x86 架构电脑不是很受重视的原因，就是因为 x86 的晶片对于多任务的处理不佳，CPU 在不同的工作之间切换不是很顺畅。但是这个情况在 386 电脑推出后，有很大的改善。托瓦兹在得知新的 386 晶片的相关资讯后，他认为，以性能价格比的观点来看，Intel 的 386 相当的便宜，所以在性能上也就稍微可以将就将就。

# Torvalds 的 Linux 发展 (2/3)——对 386 硬件的多任务测试

最终他就贷款去买了一部 Intel 的 386 来玩。早期的电脑效能没有现在这么好，所以压榨电脑效能就成了工程师的一项癖好！托瓦兹本人早期是玩组合语言的，组合语言对于硬件有很密切的关系，托瓦兹自己也说：『我始终是个性能癖』。为了彻底发挥 386 的效能，于是托瓦兹花了不少时间在测试 386 机器上！

他的重要测试就是在测试 386 的多功效能。首先，他写了三个小程序，一个程序会持续输出 A、一个会持续输出 B，最后一个会将两个程序进行切换。他将三个程序同时执行，结果，他看到萤幕上很顺利的一直出现 ABABAB..... 他知道，他成功了！

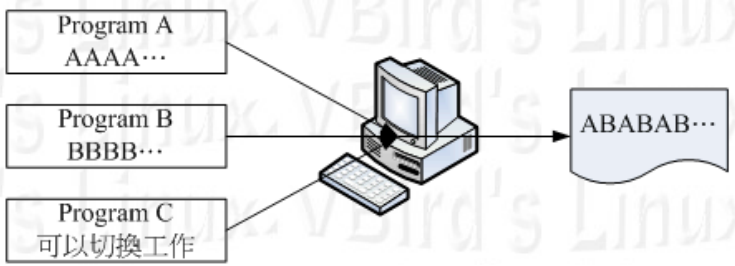


Figure: 386 电脑的多任务测试

# Torvalds 的 Linux 发展 (2/3)——初次释放 Linux 0.02

拿到 Minix 并且安装在托瓦兹的 386 电脑上之后，托瓦兹跟 BBS 上面一堆工程师一样，他发现 Minix 虽然真的很棒，但是谭宁邦教授就是不愿意进行功能的加强，导致一堆工程师在作业系统功能上面的欲求不满！这个时候年轻的托瓦兹就想：『既然如此，那我何不自己来改写一个我想要的作业系统？』于是他就开始了核心程序的撰写了。

撰写程序需要什么呢？首先需要的是能够进行工作的环境，再来则是可以将原始码编译成为可执行档的编译器。好在有 GNU 计划提供的 bash 工作环境软件以及 gcc 编译器等自由软件，让托瓦兹得以顺利的撰写核心程序。他参考 Minix 的设计理念与书上的程序码，然后仔细研究出 386 个人电脑的效能最佳化，然后使用 GNU 的自由软件将核心程序码与 386 紧紧的结合在一起，最终写出他所需要的核心程序。而这个小玩意竟然真的可以在 386 上面顺利的跑起来～还可以读取 Minix 的档案系统。真是太好了！不过还不够，他希望这个程序可以获得大家的一些修改建议，于是他便将这个核心放置在网路上提供大家下载，同时在 BBS 上面贴了一则消息：

```
Hello everybody out there using minix-
I'm doing a (free) operation system (just a hobby,
won't be big and professional like gnu) for 386(486) AT clones.

I've currently ported bash (1.08) and gcc (1.40),
and things seem to work. This implies that i'll get
something practical within a few months, and I'd like to know
what features most people want. Any suggestions are welcome,
but I won't promise I'll implement them :-)
```

# Torvalds 的 Linux 发展 (2/3)——初次释放 Linux 0.02

同时，为了让自己的 Linux 能够相容于 Unix 系统，于是托瓦兹开始将一些能够在 Unix 上面运作的软件拿来在 Linux 上面跑。不过，他发现到有很多的软件无法在 Linux 这个核心上运作。这个时候他有两种作法，一种是修改软件，让该软件可以在 Linux 上跑，另一种则是修改 Linux，让 Linux 符合软件能够运作的规范！由于 Linux 希望能够相容于 Unix，于是托瓦兹选择了第二个作法『修改 Linux』！为了让所有的软件都可以在 Linux 上执行，于是托瓦兹开始参考标准的 POSIX 规范[13]

这个正确的决定让 Linux 很容易就与 Unix 相容共享互有的软件了,linux 系统一夜之间变拥有了数以万记的 unix 软件！同时，因为 Linux 直接放置在网路下，提供大家下载，所以在流通的速度上相当的快！导致 Linux 的使用率大增！这些都是造成 Linux 大受欢迎的几个重要因素！

---

[13] Tips:POSIX 是可携式作业系统介面 (Portable Operating System Interface) 的缩写，重点在规范核心与应用程序之间的介面，这是由美国电器与电子工程师学会 (IEEE) 所发布的一项标准喔！

# Torvalds 的 Linux 发展 (3/3)——广大骇客志工加入阶段

Linux 虽然是托瓦兹发明的，不过，如果单靠托瓦兹自己一个人的话，那么 Linux 要茁壮实在很困难～因为一个人的力量是很有限的。例如托瓦兹总是有些硬件无法取得的啊，那么他当然无法帮助进行驱动程序的撰写与相关软件的改良。这个时候，就会有些志工跳出来说：『这个硬件我有，我来帮忙写相关的驱动程序。』

因为 Linux 的核心是 Open Source 的，骇客志工们很容易就能够跟随 Linux 的原本设计架构，并且写出相容的驱动程序或者软件。志工们写完的驱动程序与软件托瓦兹是如何看待的呢？首先，他将该驱动程序/软件带入核心中，并且加以测试。只要测试可以运行，并且没有什么主要的大问题，那么他就会很乐意的将志工们写的程序码加入核心中！

总之，托瓦兹是个很务实的人，对于 Linux 核心所欠缺的项目，他总是『先求有且能跑，再求进一步改良』的心态！这让 Linux 使用者与志工得到相当大的鼓励！

因为 Linux 的进步太快了！结果不到一个星期推出的新版 Linux 就有了！这不得不让人佩服！

# Torvalds 的 Linux 发展 (3/3)——广大骇客志工加入阶段

后来，因为 Linux 核心加入了太多的功能，光靠托瓦兹一个人进行核心的实际测试并加入核心原始程序实在太费力～结果，就有很多的朋友跳出来帮忙！例如考克斯 (Alan Cox)、与崔迪 (Stephen Tweedie) 等等，这些重要的副手会先将来自志工们的修补程序或者新功能的程序码进行测试，并且结果上传给托瓦兹看，让托瓦兹作最后核心加入的原始码的选择与整并！这个分层负责的结果，让 Linux 的发展更加的容易！

特别值得注意的是，这些托瓦兹的 Linux 发展副手，以及自愿传送修补程序的骇客志工，其实都没有见过面，而且彼此在地球的各个角落，大家群策群力的共同发展出现今的 Linux ，我们称这群人为虚拟团队！而为了虚拟团队资料的传输，于是 Linux 便成立的核心网站：http://www.kernel.org！

# Torvalds 的 Linux 发展 (3/3)——linux 正式版发布

而这群素未谋面的虚拟团队们，在 1994 年终于完成的 Linux 的核心正式版！version 1.0。这一版同时还加入了 X Window System 的支援呢！更于 1996 年完成了 2.0 版。此外，托瓦兹指明了企鹅为 Linux 的吉祥物。

Tips: 奇怪的是，托瓦兹是因为小时候去动物园被企鹅咬了一口念念不忘，而正式的 2.0 推出时，大家要他想一个吉祥物。他在想也想不到什么动物的情况下，就将这个念念不忘的企鹅当成了 Linux 的吉祥物了……

# Torvalds 的 Linux 发展 (3/3)——linux 正式版发布

Linux 是托瓦兹针对 386 写的，跟 386 硬件的相关性很强，所以，早期的 Linux 确实是不具有移植性的。不过，大家知道 Open source 的好处就是，可以修改程序码去适合作业的环境。因此，在 1994 年以后，Linux 便被开发到很多的硬件上面去了！目前除了 x86 之□□外，IBM、HP、Sun 等等公司出的硬件也都被 Linux 所支持！

Linux 的出现让 GNU 计划放下了心里的一块大石头，因为 GNU 一直以来就是缺乏了核心操作系统，导致他们的 GNU 自由软件只能在其他的 Unix 上面跑。既然目前有 Linux 出现了，且 Linux 也用了很多的 GNU 相关软件，所以 Stallman 认为 Linux 的全名应该称之为 GNU/Linux 呢！

# Torvalds 的 Linux 发展 (3/3)——linux 正式版发布

Linux 是托瓦兹针对 386 写的，跟 386 硬件的相关性很强，所以，早期的 Linux 确实是不具有移植性的。不过，大家知道 Open source 的好处就是，可以修改程序码去适合作业的环境。因此，在 1994 年以后，Linux 便被开发到很多的硬件上面去了！目前除了 x86 之□□外，IBM、HP、Sun 等等公司出的硬件也都被 Linux 所支持！

Linux 的出现让 GNU 计划放下了心里的一块大石头，因为 GNU 一直以来就是缺乏了核心操作系统，导致他们的 GNU 自由软件只能在其他的 Unix 上面跑。既然目前有 Linux 出现了，且 Linux 也用了很多的 GNU 相关软件，所以 Stallman 认为 Linux 的全名应该称之为 GNU/Linux 呢！

返回

# Torvalds 的 Linux 发展 (3/3)——Linus Torvalds[14]

全球最著名的 16 位黑客列传之 Linus Torvalds（李纳斯 · 托沃兹）

李纳斯 · 托沃兹——历史上最天才和受人尊敬的黑客之一，现在为 Transmeta（一家开发基于软件的微处理器的公司）工作。他已婚，有两女。

出名原因：古典视角下的真正黑客，当他在 1991 年写 Linux 操作系统（Linus's Minix 的缩写）时，他还是芬兰赫尔辛基大学的计算机科学的学生。这个操作系统很快在世界范围内受到欢迎，最好的一点是，它是完全自由的。托沃兹谦虚地把 Linux 的成功归功于网络和理查德 · 斯托曼：两者都为软件开发人员对 Linux 的开发提供了极大的方便。

[14]参考：http://www.techcn.com.cn/index.php?doc-view-130694.html

# Linux 的各种发行版本[15]

详见 linux 各发行版说明.pdf

[15]参考：http://ora110.itpub.net/post/33978/444578

# 结束语

**Thank you!**

A & Qs?

Email: lizhihua2000@gamil.com

# 结束语

# Thank you!

A & Qs?

Email: lizhihua2000@gamil.com

返回