本文衍生自維基百科

程式人《十分鐘系列》

用十分鐘瞭解

《開放原始碼的世界》

陳鍾誠

2016 年 9 月 10 日

開放原始碼

對於大部分人而言

就是有一堆軟體

- 可以讓你免費使用！

但是

- 這樣的認知

  純粹是軟體使用者的層次！

開放原始碼

- 英文是 Open Source

開源軟體

- 英文可以用 Open Source Software
- 但也有人喜歡用 Free software 來稱呼

對於程式人而言

Free 這個詞

- 絕對不是《免費》
- 而是《自由》！

就是《自由軟體》

- 裡面說的那種自由！

但是為甚麼

- 軟體需要自由呢？

關於這點

- 其實是搞錯對象！

想要自由的

- 不是軟體

因為軟體

- 不會知道自己到底自不自由

只有人

- 才會想要讓自己自由！

問題是

- 《自由軟體》到底讓誰自由！

答案很簡單

就是讓程式人自由

問題是

- 可以讓程式人得到甚麼自由呢？

關於這點

- 必須要深入的解釋一下！

想要知道自由到底是甚麼？

- 必須先能體會甚麼是《不自由》

你知道

- 我們程式人每天上班

辛苦的寫程式

背後的原因

- 其實通常和大家一樣！

那就是

老闆有發薪水！

然後

我們還領了那些錢

領了錢

就必須辦事

辦甚麼事？

老闆交代的事！

問題是

- 老闆會交代甚麼事呢？

這個問題

答案又很簡單

那些事

- 一定是老闆自己不想去做的事！

所以

- 老闆需要一些奴隸來幫他完成工作！

而程式人的工作

- 通常就是完成那些

老闆想要

- 但是自己卻不想做的程式！

換言之

程式人的工作

就是當老闆的奴隸

- 寫老闆想要的程式 . . .

身為奴隸

- 當然就沒有甚麼自由可言！

因為

- 你寫出來的程式

是公司的

而不是自己的！

於是

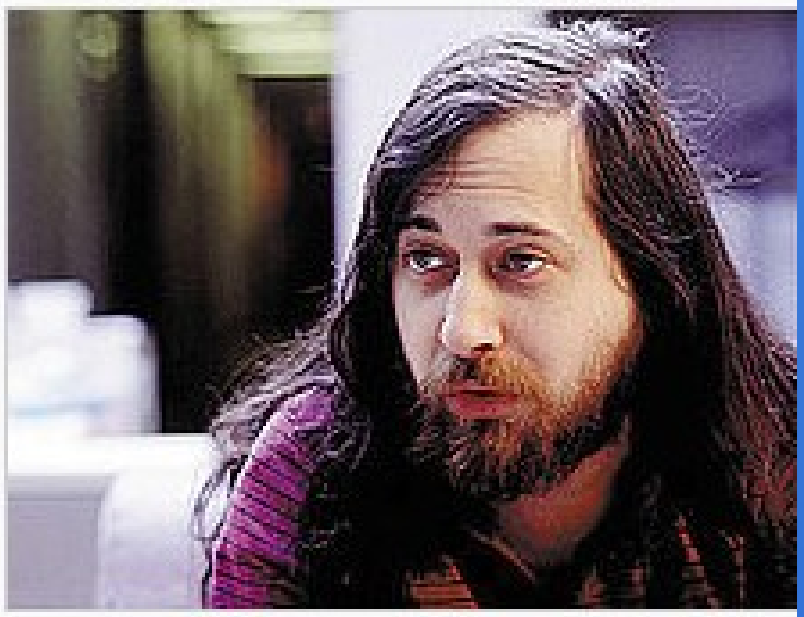
- 在 198x 年的某一天！

有一個

- 寫了很多程式的奴隸

他的名字是

- Richard Stallman



理查·斯托曼，照片來自《Free as in Freedom:理查·斯托曼的自由軟體志業》一書的封面。此書由山姆·威廉斯（2002年）著，歐萊禮出版

翻譯成中文就是

- 《理查、死拖慢》！

基本上

- 他們一群人，寫了當初 UNIX 系統裡的大部分工具程式。

但是有一天

- UNIX 的老闆 AT&T 說

這些程式

- 通通都是我的

如果你們拿去隨便用

那我就會

告你

- 告你
- 告你
- 告死你！

於是

- 《死拖慢》不爽了！

就在他很不爽的時候

發生了一件事情

那就是

他們的印表機壞了

那台印表機

- 一直印、一直印、一直印
- 都停不下來！

那時候

- 印表機還很貴！

於是

- 他打電話去向印表機公司要那台印表機的原始碼！

結果、印表機公司說

印表機的原始碼

- 是我們公司的智慧財產不能隨便給你！

死拖慢說

- 那我們那台故障的印表機怎麼辦？

於是對方說

- 請你和我們的維修部門聯絡

  你的印表機壞了！

問題是

- 出狀況的是軟體

  而不是硬體阿！

死拖慢心想

- 這家該死的印表機公司

這個該死的產業

這個該死的國家

這個該死的世界！

於是

死拖慢終於決定

要搞一場革命

革誰的命

革那些

- 軟體公司的命！

他找了律師

- 擬了一份《法律聲明》

這份聲明

- 稱為 General Public License，簡稱 GPL！

GPL 授權聲明上寫著

- 我這份程式，可以免費給你用！

但是假如你的程式

- 呼叫了我的程式

那麼

- 你的程式也必須要採用 GPL 的方式授權

於是

這場稱為開放原始碼運動的革命

- 就被啟動了！

死拖慢開始整理

- 自己所寫過的那些程式

然後分享給朋友們用

朋友們又寫了更多程式

分享給更多朋友

接著

- 這些採用 GPL 授權的程式越來越多！

像是 gcc 編譯器

- 就是這群程式裡，最為人所知的一組！

於是程式人

- 又開始得到自由了！

但是

- 還有一些小問題！

那就是

- 死拖慢他們當年

  都曾經做過貢獻的 UNIX

  現在被 AT&T 的邪惡老闆收回去了！

而且那個可惡的老闆

- 還告了他那些把 UNIX 改版為 BSD 並開放的朋友們！

https://en.wikipedia.org/wiki/Berkeley_Software_Distribution

是可忍

- 孰不可忍！

但是

- 最後還是要忍！

因為你知道

法律

- 都是站在有錢人的那邊

而不是正義的這邊！

所以

- 死拖慢决定

那我們就

- 自己來寫一個作業系統好了！

而且

- 不要和 UNIX 有任何授權上的糾葛！

所以他決定發起

- 一個叫 GNU Hurd 的計畫

  寫一個《自由的作業系統》！

問題是

- 要寫一個和 UNIX 完全沒關聯的作業系統，那還真的是件很麻煩的事情！

就在這個時候

- 有一個住在芬蘭的死白目大學生出現了！
- 那個大學生叫做托瓦茲 (Linus Torvalds)

那個大學生說

- 我想在自己的 386 電腦上跑 UNIX 。

但是我找不到這種 UNIX

所以、我決定自己來寫！

但是、我只是個大學生

而且

- 沒有寫過任何作業系統！

那我要怎麼才能

- 寫出一個作業系統呢？

那個大學生沒辦法

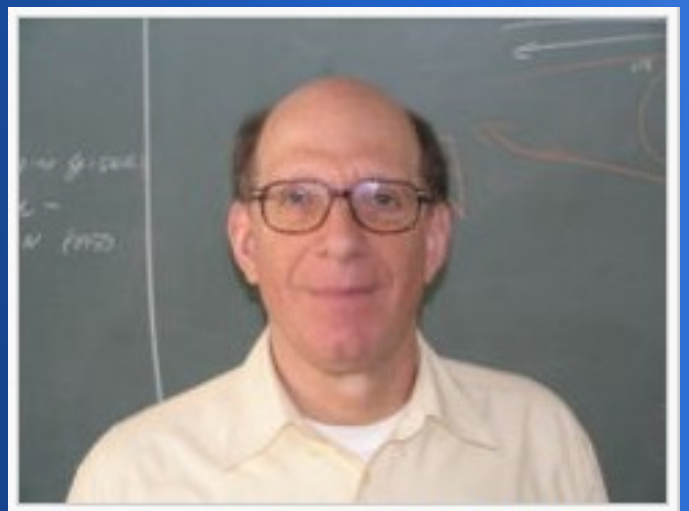
只好去找老師

他發現

- 有個叫 MINIX 的作業系統，基本上就是小型的 UNIX 。

這個 MINIX 系統

- 是由荷蘭阿姆斯特丹自由大學的 Tanenbaum 教授所寫的！



安德魯·斯圖爾特·塔能鮑姆 Andrew S. Tanenbaum

於是、那個大學生

- 很認真地讀了 Tanenbaum 教授的書，並且架起了 MINIX 來用。

然後、開始閱讀程式碼

問題是

- 托瓦茲想要的功能，很多在 MINIX 中沒有實現！

於是

- 托瓦茲決定自己來加！

加著加著

- 發現有些程式接起來不順，寫起來太囉嗦，就乾脆順手改掉了！

改著改著

- 他發現自己把《微核心》的MINIX，改回了《巨核心》的老路！

而這個巨核心版本的 UNIX

- 後來被稱為 Linux ！

更厲害的是

- 托瓦茲一邊改就一邊放上網

居然還吸引了

- 一些粉絲使用者！

這些使用者

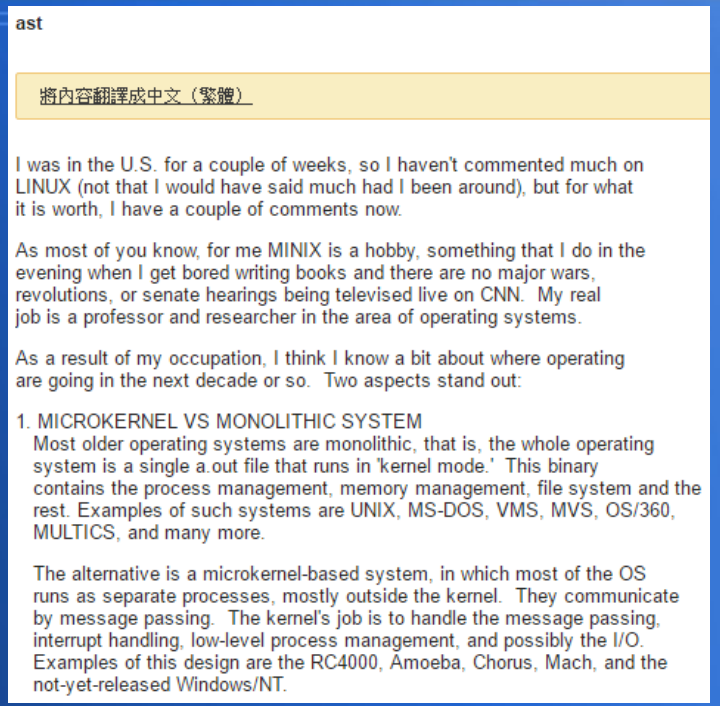
- 也大部分都是程式人，有時也會幫忙寫些功能，並傳回給托瓦茲放進去整合。

結果有一天

- Tanenbaum 教授發現了托瓦茲把 MINIX 改成巨核心的 Linux，覺得這種架構根本就是倒退十年！

於是發了訊息說

- Linux 很爛，因為不是微核心，這樣會很難擴充，很難維護！
- 還好你不是我的學生，否則一定會得到很爛的成績！



ast

將內容翻譯成中文（繁體）

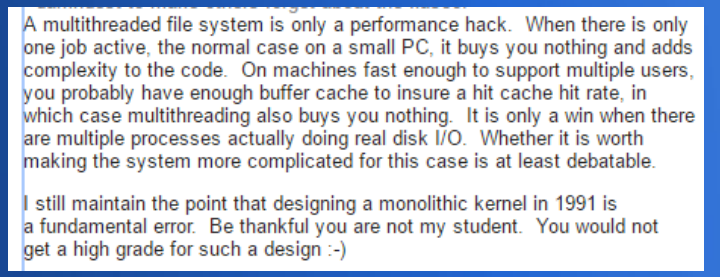
I was in the U.S. for a couple of weeks, so I haven't commented much on LINUX (not that I would have said much had I been around), but for what it is worth, I have a couple of comments now.

As most of you know, for me MINIX is a hobby, something that I do in the evening when I get bored writing books and there are no major wars, revolutions, or senate hearings being televised live on CNN. My real job is a professor and researcher in the area of operating systems.

As a result of my occupation, I think I know a bit about where operating are going in the next decade or so. Two aspects stand out:

1. MICROKERNEL VS MONOLITHIC SYSTEM
Most older operating systems are monolithic, that is, the whole operating system is a single a.out file that runs in 'kernel mode.' This binary contains the process management, memory management, file system and the rest. Examples of such systems are UNIX, MS-DOS, VMS, MVS, OS/360, MULTICS, and many more.

The alternative is a microkernel-based system, in which most of the OS runs as separate processes, mostly outside the kernel. They communicate by message passing. The kernel's job is to handle the message passing, interrupt handling, low-level process management, and possibly the I/O. Examples of this design are the RC4000, Amoeba, Chorus, Mach, and the not-yet-released Windows/NT.



A multithreaded file system is only a performance hack. When there is only one job active, the normal case on a small PC, it buys you nothing and adds complexity to the code. On machines fast enough to support multiple users, you probably have enough buffer cache to insure a hit cache hit rate, in which case multithreading also buys you nothing. It is only a win when there are multiple processes actually doing real disk I/O. Whether it is worth making the system more complicated for this case is at least debatable.

I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)

https://groups.google.com/forum/#!topic/comp.os.minix/wlhw16QWltI%5B1-25%5D

當托瓦茲看到這個訊息

開始不爽了起來！

於是開始細數 MINIX 微核心的問題

- 像是很慢，一層又一層的包裝與訊息傳遞，只是為了那個高遠的《微核心》理想，卻忘記了使用者在意的是速度，而不是你到底包裝得有多美！

https://groups.google.com/forum/#!topic/comp.os.minix/wlhw16QWltI%5B1-25%5D

還有、你 Tanenbaum 教授寫的 MINIX

- 根本就是功能不全的玩具作品，拿來玩玩可以，真的拿來用就不行了！
- 而且 MINIX 連遵循 UNIX 相容 POSIX 規格都沒有，還敢誇說移植性有多好！

這個爭吵事件

- 反映了兩人的性格！
- 一個追求完美的龜毛教授，對上一個凡事講求實際可行的工程師。

但這件事

- 只是開放原始碼發展中的一個小插曲！

追求實用的 Linux

- 終究大步甩開了 MINIX，逐漸成長為一個影響力遍及全球的作業系統！

幾年之後

- Linux 結合 Apache Server 、 MySQL 、 PHP ，成為全球 web 伺服器最愛的平台
- 但是桌上型個人市場，還是由微軟所獨佔壟斷！

有趣的是

- 在法庭上，Linux 竟然被微軟拿來當成自己沒有壟斷的證據！

當微軟因收購 Novell 被告上法庭時

- 比爾蓋茲辯稱說，我們沒有壟斷，因為伺服器上是以 Linux 為大宗 . . .

http://www.net.dicp.ac.cn/temtext/showtext.php?id=253

但是微軟

- 一邊用 Linux 來解救自己的官司
- 一邊啟動萬聖節文件計畫，開始著手準備打擊 Linux

還好

- Linux 並沒有因此被打死
- 反而逐漸的壯大起來！

又過了近十年之後

- Google 在 2007 年釋出了建構在 Linux 上的 Android 手機開發平台，後來 Android 於吃下了 80% 以上的智慧型手機市場！

現在

- 開放原始碼已經和軟體工業密不可分！

除非你是 100%
使用微軟工具的程式人

- 否則很難不去接觸到開放原始碼！

Linux 所採用的授權

- 就是而那個《死拖慢》所創造的 GPL 授權，但是加上了透過《系統呼叫》不須開源的條款！
- 這讓 Linux 上層的應用可以不用開放原始碼，排除應用程式《被強制開源》的問題！
- 這也是 Linux 得以慢慢成長茁壯的一個重要原因

除了 Android 使用 Linux 之外

- 蘋果 iMac 作業系統 Mac OS 也是以 FreeBSD 做為系統核心的。

早期 Apple 採用了 gcc 工具

- 後來卻發現 GNU 組織常常不想處理 Apple 在 Objective C 語言上的需求。
- 於是後來 Apple 開始扶植另一個以 BSD 條款授權的 LLVM 編譯器
- 現在 LLVM 也已經可以和 gcc 匹敵了！

開源的世界

- 從 1980 年代 GPL 開始算起，已經過了三十幾年！

現在開源的世界

- 非常的眼花撩亂！

因為開源專案有數百萬個

- 隨時你都可以自己加一個！

甚至每個程式語言

- 都會有自己的開源社群
- 以及開源軟體發布方法

舉例而言

- JavaScript 在 node. js 出現之後就開始蓬勃發展。
- 透過 node. js 的專案管理發布套件 npm ，我們可以輕易的對全世界發布專案，並設定授權方式！

就算不透過 npm 這種工具

- 我們也可以用《托瓦茲》創造的 git 工具，輕易的將專案發布在像 github 這樣的開源平台上。

於是現在的程式人

- 面對的是一個五花撩亂的世界

我們得從

- 數百萬個專案裏挑出自己要用的套件，然後安裝、學習、開發、上傳、然後發布自己的專案！

而且還要注意到

- 專案的授權與相容性等問題！

像是

- 如果你用了 GPL 的套件就得要用 GPL 授權釋出自己的專案！
- 但是若用 BSD 授權的套件，則只需要標示該套件的 BSD 授權就行了！

另外還有像

- Mozilla, Apache, CC, MIT ... 等各種授權方式，您必須仔細分辨，以免誤用授權而導致違反授權的問題。

現在的程式人

- 感覺比較像處在一個，充滿各種《不同自由》的自由世界當中，而且必須要小心的不要超越這些自由！

當你懂得這些自由的限制

- 才能真正的享受用他人的程式來組合並開發自己程式的自由！

如果你懂得使用開源套件

- 或許會逐漸發現

  寫程式不再是一種手工業！

而比較像是

- 一種組裝業

程式人的責任

- 逐漸轉變成從數百萬的專案中，挑出對自己有用的東西，然後想辦法《組裝》使用
- 我們只要用很少得程式碼將這些專案黏合起來就好了！

當然

- 不是每個領域都有適合的專案，也不是每個專案都可以完全符合使用
- 所以必要時我們得挖開專案內部，進行修改或重新打造！

這就是

- 我這個程式人，面對現在這個五花八門世界時的看法！

希望

- 你也能在開源的世界裡

得到真正的

自由！

這就是我們今天的

十分鐘系列

我們下回見！

Bye Bye!