Linux 的特色

何浩洋
國立暨南國際大學資訊工程系
Email: u7321044@ncnu.edu.tw

姜美玲
國立暨南國際大學資訊管理系
Email: joanna@ncnu.edu.tw

摘要

我們要怎麼樣去看待 Linux 的核心程式 (kernel) 呢？要如何下手呢？在本篇文章，我們先從一個應用程式的角度去看它，然後再去細看它的核心程式架構，接著我們會分析 Linux 的程式設計特點，就 Linux 在穩定、安全、速度、簡潔、相容性，以及可移植性 (portablity) 等方面做分析及討論。總結，Linux 的核心程式雖然是歸屬於巨大核心 (Monolithic Kernel) 之類，然而 Linux 藉由使用了模組 (Module) 的機制及抽象化階層 (Abstract Layer) 設計的觀念，使其具備了 Monolithic Kernel 與微核心 (Micro-kernel) 的優點。

1. 應用程式的角度

當一個應用程式在執行時，會透過系統呼叫 (System call) 去呼叫核心的服務 System call 是由 System call interface 去管理。如果應用程式要利用某些電腦資源例如：記憶體，便會涉及記憶管理 (Memory Management) ，核心的記憶體管理的工作就是管理及分配記憶體的位置及空間。

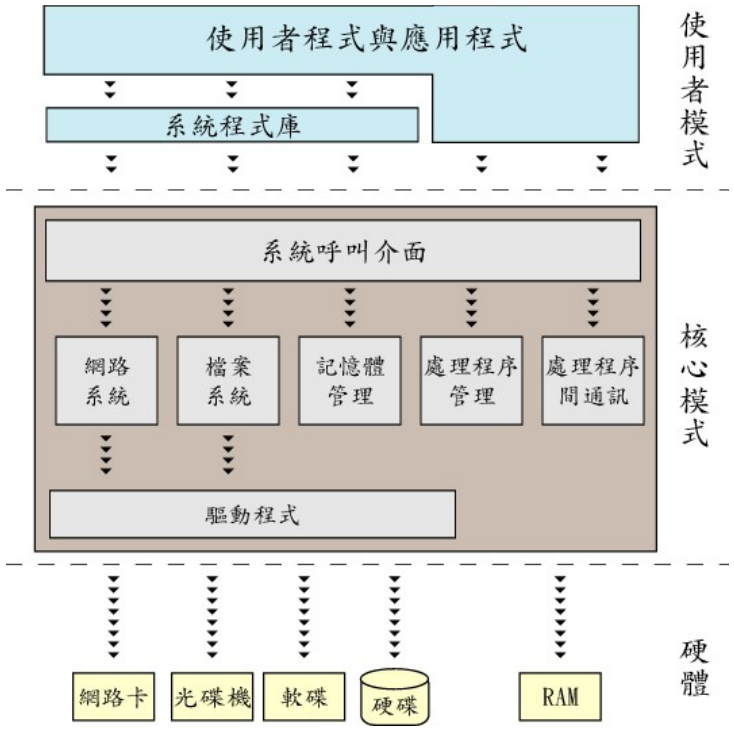
而應用程式若需要檔案 (Files)，便會涉及檔案系統 (File System)，而核心的檔案系統就是負責管理及分配檔案的位置及空間的工作，當應用程式需要網路的功能，便涉及網路管理 (Network Management)，核心的網路管理的主要工作就是管理封包的傳送與接收，而 CPU 資源，就由核心的處理程序管理 (Process Management) 去管理每一個處理程序 (Process) 所分配到的時間以及執行順序。

除了上述的資源分配管理之外，還有處理程序間通訊 (Inter Process Communication, IPC) 以及啓動程序(Boot) 兩個部分。IPC 是管理 Process 互相溝

通的橋樑，而 Boot 則是在 kernel 載入前的時候就已經結束，雖然從應用程式的角度看不到，但要是沒有 Boot 這一個部分將 kernel 載入，kernel 便無法啟動。

2. 核心程式架構

Linux 的核心程式架構如圖一，它包含了以下的各部份:



圖一: Linux 的核心程式架構

- 記憶體管理 (Memory Management)
  提供虛擬記憶體，管理及分配記憶體以滿足處理程序執行時的需求。

- 處理程序管理 (Process Management)
  管理處理程序的執行，如處理程序的建立 (create)、終結 (terminate)、等候 (wait)、本文交換 (context switch)，以及提供不同的排程方式 (scheduling)。

- 處理程序間通訊 (Interprocess Communication)
  提供多種不同的程序間通訊的機制。

- 檔案系統 (File System)
  提供一致性的階層式檔案系統 (Uniform Hierarchical File System) ，透過抽象化的虛擬檔案系統 (Virtual File System) ，支援多種不同類型的檔案系統。

- 網路管理 (Network Management)
  支援多種不同的通訊協定，管理封包的傳送與接收。

- 設備控制與管理 (Device Control and Management)
  提供多種不同硬體設備的驅動程式，以及設備的控制與管理

3. Linux 的程式設計特點

在我們介紹 Linux 設計上的特點之前，先來了解一些程式設計上的根本觀念這樣才能了解 Linux 為何會被如此設計出來。

在設計作業系統時，程式設計要考量的地方不外是：穩定、安全、速度、簡潔、相容性，以及可移植性 (portablity)。大家一般可能會說速度是最重要的，但是這個卻不是絕對的，尤其對一個作業系統而言，穩定和安全才是最重要的。這是為什麼呢？很簡單，如果你的作業系統開不了多久便當機，不然就是很容易被他人入侵，那速度再多快也沒有什麼意義了。然而速度通常會跟簡潔有所衝突。為什麼這樣說呢？不是越簡潔越快嗎？在這裡的簡潔是指程式碼的寫法。用最簡單直接的方法去實作演算法會比用複雜的方法來得容易明嘹，但執行效率就可能會有所差距了。

在 Linux 底下，速度比簡潔來得更重要，所以為了加快速度，只好捨棄簡潔的寫法，用一些比較複雜的寫法去提昇速度；所以要從程式碼去了解為什麼要這樣寫，實在是蠻困難的一件事。而簡潔的寫法，跟穩定和安全有很大的關係，因為寫法越簡潔，找出錯誤就越簡單，且一看就很容易找出問題所在。

而相容性，有時候會比速度還要重要。因為可以支援的東西越多，它的應用範圍也就越廣，更多人就可以使用。Linux 在這方面做的很好，它支援很多不同的檔案系統和很多不同的網路通訊協定，相容性十分高，至於它是怎樣做到的，在後面的章節我們會再深入研究。

而可移植性，是指程式若要從一個平台移植到其他平台是否容易的問題，如果可攜性越高，移植到其他平台便越簡單，你可以在 Intel x86、PowerPC、Apple、Alpha 等平台都可以看到不同套件(distribution)的 Linux，由

此可想 Linux 的可攜性之高。

簡單的介紹過以上幾點之後，我們來看看 Linux 是怎樣的一個作業系統，它的 kernel 是怎麼樣達到以上幾點。

在穩定和安全方面，Linux 有分核心模式 (kernel mode) 和使用者模式 (user mode) 二個模式，kernel mode 是處理 kernel 的工作，例如資源分配等等；在 user mode 模式，不同的使用者各有自己的權限，不會影響別人的資源，當然，檔案目錄等等也有其權限的設計。

在速度方面，它有點放棄了簡潔的方法，用了很多指標 (pointer)，和特別的資料結構，加上 Linux 本身對系統資源的分配上有最佳化的策略，使系統常常保持最佳化。

在簡潔方面，Linux 為了加快速度而寫得比較複雜，所以寫 Linux 核心的高手們都常都會在重點地方加上註解。

在相容性方面，Linux 核心提供了很多不同檔案格式的支援，例如：Ext2、vfat、proc、、、等等，還有支援很多不同的硬體架構和設備，也支援多種不同的網路協定。

Linux 利用 module 來實作這些不同的支援，當你有需要使用某一些功能的時候，就可以把某些 module 載入核心，不用時，就可以移出核心。

在可移植性方面，一個核心除了可以從 Process 的角度去區分，也可以大略區分成跟硬體有關（Hardware Dependent）和跟硬體無關（Hardware Independent）兩大部份。Linux 要從一個平台移植到不同的平台，為了減少對程式碼的修改，Linux 把核心分成跟硬體有關和跟硬體無關兩部份，方便移植，這點從程式碼放置的目錄就可以看出，與硬體相關的程式碼皆被放置在 linux/arch 目錄底下。

Linux 在這兩層之間定了一層很清楚嚴謹的介面，讓兩方面去實作，這是為了減少因修改跟硬體有關的程式碼而影響跟硬體無關的部分。Hardware independent 是不會理會 hardware dependent 的部份，因為 hardware independent 使用的是已經定好的介面；而 hardware dependent 也是一樣，當如果要更動硬體架構只要實作出硬體相關的原始碼來就好了。如此一來，修改的工作就變少了，從而提高 Linux 的可攜性。

4. Linux 擷取 Monolithic Kernel 跟 Micro-kernel 的優點

如果手提電腦分成 all-in-one 跟超薄形的話，那麼 kernel 也能分成巨大的核心（Monolithic kernel）和微小的核心（micro kernel）。Monolithic kernel，它本身就是把很多的功能都收集到核心之中，使核心的功能很強，又因為有很多功能都在核心之中，所以它們之間的溝通都是很直接地相互呼叫使用，不過，有一利就有一弊，如果我們把所有的功能都放進核心之中，會使修改核心程式碼的工作變得十分困難且複雜，所以會降低它的可攜性，又因大量的功能加入核心之中，會使核心的體積變得很龐大、花費更多的記憶體。

因為這樣，所以有 micro kernel 這另外一種設計的概念發生，它的設計剛好跟 Monolithic kernel 相反，盡量讓 kernel 跟作業系統的其他功能分開，把 kernel 獨立出來，使更容易去維護，也正因為如此，它需要透過如 Message Passing 的方式去使用作業系統的其他功能，而不能像 Monolithic 核心般可以直接呼叫，這樣一來，雖然它的體積比 Monolithic 核心小，不過它所執行時的 overhead 比 Monolithic 核心來得重。

因此 Linux 核心走的是 Monolithic kernel 的實作改良版，而不是純粹的 Monolithic。它對 Monolithic 核心的問題作了一些改變，Linux 吸收了 micro 核心的一些優點去補救 Monolithic 的問題，它使用了模組 (Module) 的機制，在需要的時候才動態地載入核心使核心盡量減肥，不必把所有的 module 都載入記憶體；且使用 Layer 的觀念，採用 virtual layer 去降低更改硬體時對核心的影響，在 Monolithic 跟 micro kernel 兩者中取得其優點。

總結，Linux 核心並不是由証明理論而產生的。它是由全球不同的 hacker 經過平日使用修改而產生出來，所以它的設計十分現實，從它身上可以找到很多不同理論的特質。

5. 參考文獻

[1] Michael Beck, Harald Bohme, Mirko Dziadzka, Ulrich Kunitz, Robert Magnus, Harold Bohme, Linux Kernel Internals 2nd Edition, ISBN 0-201-33143-8, 1999.

[2] Scott E. Maxwell, Linux Core Kernel Commentary, ISBN 1-576-10469-9, 1999.

[3] The Linux Kernel Archives, http://www.kernel.org.