# 縮減 Linux 嵌入式系統軟體: 方法與實例

梁元彪　林盈達
國立交通大學資訊科學系
300 新竹市大學路 1001 號
Tel: 03-5712121 ext. 56667
Fax: 03-5712121 ext. 59263
{upleong,ydlin}@cis.nctu.edu.tw
聯絡人: 梁元彪

## 摘要

開放式原始碼的迅速發展，正逐步滲透至嵌入式系統，兩者的結合將成為後個人電腦時代中極為重要的一環。 首先我們在此探討使用 Linux 作為嵌入式作業系統的方法，透過如何對核心 (kernel)，守護神程式 (daemons)，程式庫 (libraries) 和應用程式 (applications / utilities) 等四個主要部份，進行縮減其大小後，以便配置在以快閃記憶體為儲存設備的嵌入式系統中。 接著以提供 VPN，防火牆，入侵偵測系統 (IDS) 等功能的安全閘道器[1] 作為例子，說明如何建立一個完整的嵌入式 Linux 系統。 最後我們把擁有以上所有功能的安全閘道器，從使用 Redhat 的系統 (約 1.3 GBytes) 以及額外的 RPM 套件 (約 15 Mbytes)，成功縮減系統至 19 Mbytes。 經壓縮為 7 Mbytes 後，置於配備 64M 記憶體與 8M 快閃記憶體的系統中。

**關鍵字:** embedded, Linux, downsizing

## 1. 背景

嵌入式系統 (embedded system) 已在無形中，慢慢融入我們日常生活裡。 各式各樣的數位相機，個人數位助理，寬頻上網用的 ADSL 或纜線數據機等，全都依賴著程式的控制。 在這些的例子中，我們可以發現，它們都擁有特定的功能，運作穩定而且非常普遍。 這就是嵌入式系統的威力，使用的資源少，穩定度高，價格低。 挾著眾多的優點，嵌入式系統將成為後個人電腦時代的重要產品技術。 在表 1 中，列舉了一般個人電腦與嵌入式系統在資源與功能上的差異。 嵌入式系統所用的儲存設備為快閃記憶體，比硬碟耐用但容量很小。 其次會省去 VGA 等不必要的設備。 最主要的是在功能上，嵌入式系統的功能特定，所以能對其最佳化，也易於進行測試，並且大大提高軟體在系統中的穩定度。

**表 1. 個人電腦與嵌入式系統的差異**

| | Personal Computer | Embedded System |
|---|---|---|
| CPU | Pentium | Pentium / StrongARM / PPC / x86 SOC |
| RAM | 128M or above | 8M~64M |
| Storage | HD (~GB) | Flash (2M~16M) |
| Ethernet | NIC | Built-in |
| Console | VGA | Serial |
| Purpose | General | Specific |
| Stability | Fair | Good |
| Cost | High (~US 300) | Low (~US 150) |

就網路設備而言，功能全面，穩定與互通性等因素，是極重要的關鍵。 在開放式軟體蓬勃發展以前，市場多為商業軟體所佔有，直至 Linux 等作業系統的崛起，在網路領域上已成為極佳的選擇之一。 表 2 對 Linux 與各廠商提供的作業系統做了一個簡單的比較。 我們可以看到商業軟體的價格高昂，並且有收取權利金的制度。 另外每家廠商提供的軟體大多並不相容，作業系統也屬於封閉式，所以一般程式對其支援比較缺乏，嚴重依賴廠商的提供。 嵌入式 Linux 提供了原始碼，與一般 UNIX 系統的應用程式相容，以及完全自由取得等優勢，皆對各歷史悠久的系統造成威脅。 然而，正因為還在起步階段，目前的嵌入式 Linux 版本並算不上是一套非常簡潔的系統。 如何善用與分配資源，整合一般 Unix 的程式，以及縮減系統至適合嵌入式應用成為最主要的目標。

表 2. Linux 與商業嵌入式作業系統之比較

| | Embedded Linux | VxWorks | QNX | pSOS | Windows CE |
|---|---|---|---|---|---|
| Sales | Free | $130M ($20M royalty) | $20M | $105M ($15M royalty) | $3M royalty (FY99) $2M Tools |
| Market Share | N/A | 15% | 2% | 12% | <2% |
| Tools Price (per seat) | Free or from 3rd party | $7,500-$10,000 (Based on a 10 seats license) | $2,995 per seat (3rd party tools) | $4,995 by seat Depends on the options | $995 |
| Royalty | No royalty | $1-$35, no real fixed price for licensing | Component-based pricing $3-$80 | Range from <$1 up to $30 | Range from <$5 up to $30+ |
| Focus | Networking Datacom | Digital imaging, telecom/datacom, CE, automotive | Industrial automation, medical systems, CE, telecom/datacom | Automobile industry, some CE (interactive television) | Higher-end apps; ADCU has focused on Manufacturing, Retail, Healthcare |
| Strengths | 1. Open source<br>2. Third party supported | 1. Most complete integrated environment<br>2. Largest coverage of CPU and hosts | 1. True micro-kernel architecture<br>2. Complete POSIX compliant<br>3. Most scalable | 1. Integrated development environment<br>2. Optimized for Motorola processors and DSPs | 1. Win32<br>2. NT Connectivity<br>3. Robust graphics/UI |

接下來先了解具有高度彈性的 Linux，有那些主要可被縮減的部份。 再分析縮減至嵌入式版本的方法及其開發工具。 在本文的最後，以安全閘道器為例子，介紹其功能規格和所使用的套件，並呈現經過上述縮減過程並移植至嵌入式系統後，其顯著的成效。

## 2. 可縮減的部份

Linux 軟體不斷的發展，累積越來越多的程式碼，形成相同功能程式有多種選擇的優勢。 而我們需要的是一個精簡的系統，因此必須進行縮減。 首先把龐大的系統歸納為四個部份 -- 核心 (kernel), 守護神程式 (daemons), 程式庫 (libraries), 應用和工具程式 (applications and utilities)。 我們在進行縮減的原則是在不直接刪改程式碼，以保有原始碼的完整性。 表 3 列出縮減各子系統的方向與目標，以下為詳細的介紹。

表 3. Linux 中可以被縮減的部份

| | 進行縮減的方向 | 縮減的方式 |
|---|---|---|
| 核心 | 使用固定的硬體 | 自訂目標系統的 Linux 核心 |
| 守護神程式 | 提供特定的服務 | 在編譯時只設定所需之功能 |
| 程式庫 | 去掉沒被用到的共用程式庫 | 檢查相依性並只保留有被使用的 |
| 應用和工具程式 | 簡化複雜的功能 | 以專為嵌入式系統設計的程式取代 |

### 2.1 核心

因為嵌入式系統的功能規格定義明確，在核心的設定中只須留下必要的選項或模組等，其他不必要的都可以捨棄。 另一種就是利用取代的方式。 如果是網路設備，可以把 console 的輸出以序列埠來取代顯示卡的輸出。 藉此不單可省下硬體成本，更可在 Linux 2.4 的核心中省下 42KB 的空間。 除此以外，平行埠，隨插即用，軟碟機，光碟機，鍵盤，滑鼠, USB 等驅動程式都可以被省略。

### 2.2 守護神

在開放式原始碼的影響下，各程式除了往更完整功能邁進的同時，對各種軟硬體的支援亦漸趨多元化。 然而在有限的資源下，必須根據需求，重新定義各程式所需要的功能，而應避免安裝原本可直接執行的套件。 舉例來說，如果只使用 Squid 作為 http proxy, 而並沒用到其 caching 的功能。 就可在編譯時，直接利用 Squid 的組態設定程式，關掉對檔案系統的支援 (使用選項 --enable-storeio=null)。 這樣在 Squid 2.4.Stable1 的版本中，所得到的程式碼，便可省掉 145KB (約 26%) 之多。 其他像 GNU Zebra 這些網路相關的程式，也可以把 IPv6 的支援關掉以節省空間。

### 2.3 程式庫

程式使用靜態或動態連結，會產生不一樣的特性。 靜態連結使得程式執行時 overhead 減少，程式碼也比較簡潔。 而動態連結則隨共用程式庫的程式數目增加而節省大量的空間。 因此必需根據需求在兩者之間取得平衡，如果程式庫只有一支程式在用，便可考慮選擇使用靜態連結。 而使用動態連結的情況下，可以利用工具程式 ldd 來檢查程式與程式庫間的相依關係。 藉由此法，便可找出及保留系統所需的共用程式庫的最小集合。

### 2.4 應用和工具程式

在這個部份就具有更多的彈性了，因為這類程式的選擇性多，且可用多種方法替代。 以核心為例，透過對 /proc 的支援，便可以讀取或修改多項系統的參數設定，不必再使用累贅或不常用的工具程式。 例如只要讀取 ARP table 時，便可使用

指令 #cat /proc/net/arp 來達成，省卻使用 /sbin/arp 這支程式，完全不漸空間。 其次在嵌入式系統上，對象不再是一般 PC 使用者，簡化程式介面，線上說明，甚至簡化程式功能等，都可以實現精簡化的效果。 BusyBox[2] 就是其中具有代表性的一套工具程式，它以單一的小程式，十分精簡地提供了檔案工具，shell，文字處理，壓縮程式等 UNIX 中常用的功能。 表格 4 列出 BusyBox 與 tinylogin[3] 的功能及其相對原工具程式的大小。 由此可見充份利用專為嵌入式系統而設計的工具，是一個取得雙贏的好方法。

**表 4. 嵌入式程式功能列表**

| 嵌入式程式及其大小 | 提供之功能及其對應之各個程式大小之總和 |
|---|---|
| BusyBox (Text size 192 Kbytes) | Cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, false, fdflush, gunzip, gzip, hostname, kill, ln, ls, mkdir, mknod, mktemp, more, mount, mt, mv, ping, pwd, rm, rmdir, sed, sleep, sty, sync, tar, touch, true, umount, uname, vi, zcat (Total text size 719 Kbytes) |
| TinyLogin (Text size 53 KBytes) | Addgroup, adduser, delgroup, deluser, login, su, getty, passwd (Total text size 75 Kbytes) |

### 2.5 除錯資訊與符號表 (Symbol Table)

在除錯階段結束後，在程式內用於除錯的資訊，全部都可以被刪掉，因為它們都提供配合原始碼除錯的功能，所以佔了很大的空間，可以使用工具程式 strip 刪去。 以 IDS 的 snort daemon 和 squid daemon 為例，分別從 969K 與 670K，減至 307K 與 419K。 此以，我們也可以對所有執行檔進行掃瞄，找出在共享程式庫中沒有被用到的符號列表，將它們從程式庫使用 strip 移除，進一步達到縮減系統的目的。

## 3. 如何進行縮減

### 3.1 進行方式

目前 Linux 的發行版本 (distribution) 眾多，各有優缺點，要集各家之大成，建立一套屬於自己的嵌入式 Linux 並非夢想，只要根據適當的方法進行就可以。 首先以 Redhat 7.1 的發行版本來看，完整安裝伺服器的版本，約須 1.3 GBytes 以上的空間，這還不包含一些特別的套件。 然而，在眾多的軟體中，我們只要其中的一小部份，可能在 10~30 MBytes 之間。 進行的方式必須有效率的建立系統，基本可行的方法有兩種: (1) 從其中一個發行版本中把不必要的部份全部刪掉，留下我們想要的系統。 (2) 把系統所有的功能依規格從新建立起來。

正如前所述，我們必須先對系統功能規格清楚了解，才能進行縮減系統的動作。 從圖 1 可見，使用方法一，會有多達幾百 Mbytes 不必要的資訊在旁影響。 使用方法二，則可以很直接的建立一個從 2MB 至 16MB 大小的嵌入式系統。 所以使用重建的方法，顯然是比較好的選擇，要使用此方式前，得先準備三件事 -- 整體功能規格，使用的軟體套件，目標平台的規格。 其實這三者之間息息相關，也就是從三個不同的角度去定義系統的做法，使其更全面更準確。

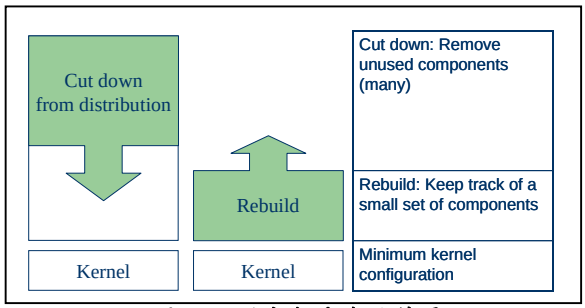


**圖 1. 刪減與重建的差異**

### 3.2 開發環境

在清楚了解準備建立的系統後，便可開始利用 Linux 打造嵌入式的設備了。 首先我們必須區分開發平台與目標平台。 目標平台就是程式最後被移植及執行的地方，因為資源有限，開發與除錯的環境都會集中開發平台上。 表 5 列出兩個平台上的主要開發工具。

**表 5. 開發與目標平台上常見之軟體功能**

| **在開發平台上的工具軟體** | |
|---|---|
| 跨平台發展工具 (e.g. arm-linux-gcc, arm-linux-gdb) | 編譯目標平台所能執行的程式碼 |
| 檔案處理程式 (e.g. mke2fs, mkcramfs) | 製作目標平台的檔案系統 |
| **嵌入式目標平台的工具程式** | |
| 開機程式 (e.g. etherboot, redboot) | 檢查系統的完整性，自我測試，透過網路或快閃記憶體開機 |
| GDB server | 透過 gdb 以序列埠或網路進行除錯功能 |

要建立一個最基本的發展環境，必需具備一套跨平台的發展工具 (Cross Development Kit)，包含有編譯器[4]，連結器，除錯器等。 另外還要準備製作檔案系統所需的程式。 而目標平台上，只需準備一段系統開機程式，如 etherboot[5], redboot[6] 等。 此程式可以在除錯階段時，從網路取得系統映像檔 (image) 後啟動，或是直接從快閃記憶體中，把系統啟動。 一旦啟動後，就進入 Linux 的作業系統，同時亦可使用 GDB server 作

為遠端除錯的工具。

### 3.3 開發過程

開發的過程如圖 2 所示，可以分為幾個部份。首先要準備 Linux 的核心，配上根目錄所在的檔案系統，再加上守護神程式和應用程式等，經過壓縮後，打包成一個含有核心的映像檔。 目標平台透過網路或快閃記憶體，取得映像檔後，進行解壓縮，經過系統啟動，初始化後，就是一台使用嵌入式 Linux 為作業系統的機器。

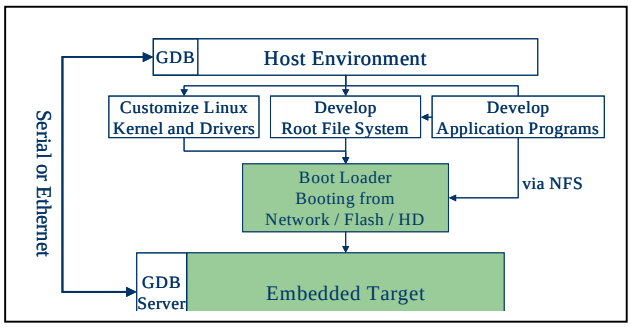


**圖 2. 系統開發流程圖**

建立檔案架構的時候，必須注意權限的設定。因為快閃記憶體的可覆寫次數有限 (約為一百萬次)，所以會以唯讀的方式掛上 (mount) 系統。 而 /var /dev 等目錄須為可讀寫的，或是記錄一些暫存檔和記錄檔等，都可以利用系統的記憶體空間來模擬可寫入的檔案系統 (RAM disk)，但。 透過建立符號化連結 (symbolic link) 取得寫入權限，餘下的則由唯讀的方式被保護在快閃記憶體之上。 如果大量的記錄檔必須在關機後留下的話，外掛硬碟至系統 /var/log 是比較適當的做法。 小量的記錄檔是可以寫入快閃記憶體中，但一定要使用緩衝記憶體，否則頻繁的寫入會使其使用壽命縮短。 同理，我們也不可能使用 swap，若然記憶體實在不足的情況下，可以考慮系統留在快閃記憶體上，同樣以外加硬碟機作為 swap disk 的功能。

在嵌入式系統中，會配合 ramdisk 使用在系統上，但是會與使用一般的檔案系統有差別，因為 ramdisk 會使用部份記憶體，因此可用的記憶體相對減少，可能造成對記憶體使用狀況的誤解，必須特別注意。 圖 3 說明了使用 ram disk 時的記憶體空間分佈狀況。

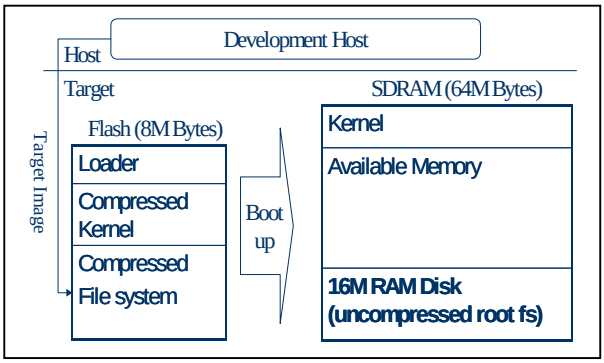


**圖 3. 映像檔在記憶體中展開的示意圖**

## 4. 安全閘道器實例

### 4.1 功能簡介

接下來我們以安全閘道器為例，說明根據上述方法發展嵌入式版本的成效。 安全閘道器是一台具有防火牆，VPN 與入侵偵測系統的閘道器。全部皆以開放式原始碼的套件組成，主要有 Linux Kernel 2.4.7[7], netfilter (packet filter, port redirect, NAT), Squid (URL filter)[8], TIS (content filter)[9], FreeS/WAN (VPN)[10], 與 Snort (IDS)[11]。 另外還有其他的功能，如 DNS, DHCP server, routing, bandwidth management[12], web-based configuration 等。

### 4.2 硬體規格

選用配備快閃記憶體的工業級電腦，CPU 為 Pentium II 350, 64MB Ram, 8MB Flash, 10/100Mbps NIC 三張。 其中程式會先壓縮後放至快閃記憶體中，所以實際系統的大小可以容許超過 8Mbytes 的實體限制。

### 4.3 使用之主要套件及處理方式

在表 6 中列出了安全閘道器的主要功能中所用到的套件。

**表 6. 安全閘道器所使用之主要套件列表**

| Category | Package | Daemon / Program | Kernel-space Program |
|---|---|---|---|
| Kernel | Linux Kernel 2.4.7 | N/A | Kernel with VPN patch |
| VPN | FreeS/WAN 1.9.1 | Pluto daemon (Internet Key Exchange, IKE) | KLIPS kernel patch (Encryption and authentication) |
| Firewall | Netfilter (packet filter) | Management tool (iptables) | Kernel built-in packet filter |
| | Squid 2.4.STABLE1 | URL filter Transparent proxy | N/A |
| | TIS 2.1 | Content filter | N/A |

| | | | |
|---|---|---|---|
| | POP3 Proxy | POP3 mail proxy | N/A |
| Address Management | Netfilter (NAT, port redirect) | Management tool | Kernel built-in IP masquerade and port forwarding |
| | ISC DHCP Server | DHCP Daemon | N/A |
| | DNS Server | DNS Daemon (named) | N/A |
| Intrusion Detection System | Snort 1.8.1 | IDS Daemon | N/A |
| Policy-based Bandwidth Management | IProute2 + tc | Management tool | Kernel QOS support required |
| Routing | GNU Zebra 0.91a | RIP, OSPF routing daemon | N/A |
| System Management | Apache httpd 1.3.19 | Web server with CGI support | N/A |
| | Perl 5 | Perl interpreter | N/A |

#### 4.3.1 核心縮減過程

將核心的原始碼解開後，進行設定 (指令 # make menuconfig 或 # make xconfig) 核心選項 (kernel options)，其中會以序列埠 (打開 CONFIG_SERIAL, CONFIG_SERIAL_CONSOLE) 取代 VGA Console (關閉 CONFIG_VT, CONFIG_{VT,VGA}_CONSOLE) 以及開機時使用記憶體作為硬碟使用 (打開 CONFIG_BLK_DEV_{RAM,INITRD})。 最後就可以編譯得到新的核心 (指令 # make dep bzImage)。

#### 4.3.2 守護神縮減過程

我們以 Squid 守護神作為例子。 因為在安全閘道器上只做透通性的 http proxy 與 URL filter 服務，除了在設定檔 (squid.conf) 中把快取功能關掉以外，可以更進一步的從程式中對快取支援的部份刪掉。 在 2.4.STABLE1 的版本中，提供了三種快取儲存格式，分別是 UFS, AUFS 和 NULL，最後一個表示不使用快取，所以符合需求。 接著使用組態程式進行設定 (指令 # ./configure --enable-storeio=null --enable-linux-netfilter) 並使用 Linux netfilter 模組，最後便可以進行編譯。

#### 4.3.3 程式庫縮減過程

這裡所指的都是對動態程式庫的縮減方法，因為共用所以要考慮的是整個系統而非單一程式庫。 透過使用 script 程式，先找出系統內執行檔的相依性 (使用 ldd, list dynamic dependencies)，再找出執行檔與程式庫間的 symbol 使用狀況 (使用指令 #objdump –T 來檢視 dynamic symbol table)，統計所得到的結果。 最後依據記錄，使用程式 strip，去除執行檔和程式庫多餘的部份。

#### 4.3.4 工具程式縮減過程

使用 BusyBox 作為工具程，以簡化的功能取代系統工具。 首先把原始碼解開，選定所需的功能，在設定檔 (busybox-0.51/Config.h) 內進行定義，如 #define BB_PING, #define BB_SLEEP, #define BB_ROUTE 等。 在列舉功能後，便可以進行編輯，而最後的指令，都以 symbolic link 的方式連結至 /bin/busybox 。

## 5. 成效之比較

在最後，依照以上所提的各種方法，分別對安全閘道器上不同的程式，進行縮減的過程。 表 7 為一些具代表性的功能或套件，分別在兩個平台上的比較，最後是縮減的比率。 以這些套件來看，可以被縮減的比率達到五成。

**表 7. 安全閘道器上各套件縮減之成效**

| 程式 | 套件內之大小 (bytes) | 嵌入式內大小 (bytes) | 縮減方法 | 改善比例 |
|---|---|---|---|---|
| Kernel | 707K | 627K | 重新組態 | 11% |
| Pluto | 1921K | 636K | Strip symbol | 67% |
| Squid | 680K | 442K | 重新組態 | 35% |
| TIS | 288K | 101K | Strip symbol | 65% |
| Snort | 1036K | 283K | 重新組態 | 73% |
| Iproute2 | 96K | 85K | Strip symbol | 11% |
| GNU Zebra | 2484K | 618K | 重新組態 | 75% |
| Apache | 1037K | 587K | 重新組態 | 43% |
| TinyLogin | 75K | 53K | 嵌入式程式 | 29% |
| BusyBox | 719K | 192K | 嵌入式程式 | 73% |
| Libraries | 3525K | 2664K | Strip symbol | 24% |
| ***Subtotal*** | ***12568K*** | ***6288K*** | ***N/A*** | ***50%*** |
| Web GUI | 3380K | 3380K | No change | N/A |
| Perl (libs) | 8694K | 3190K | Select library | 63% |
| Misc | 7938K | 7418K | Strip symbol | 7% |
| ***Subtotal*** | ***20012K*** | ***13988K*** | ***N/A*** | ***30%*** |
| ***Total*** | ***32580K*** | ***20276K*** | ***N/A*** | ***38%*** |

當中可見，核心的部份把所有未用到的驅動程式拿掉以後，會縮減 11% 的大小。 若要進一步縮

減的話，可能就要修改部份程式，或是改以模組化的形式編譯了。 另外, Pluto daemon 與 TIS 都因為有大量的除錯資訊在內，只使用 strip symbol 的方式就有不錯的成效。 而 Snort 的部份可達 73% 是因為在套件裡預設支援 mysql 的資料庫，使用重新組態以及去掉除錯資訊，就可以大幅刪減了。 而 web server 的部份也是在套件中預設的多功能，而使得重新編譯後縮減的比率較高。 在程式庫方面，則是因為已是使用的最小集合，其縮減空間只是一些未被用到的 symbol，所以只有約 24% 的縮減比率。 在最後列出的部份，包括未經處理和未分類的程式部份。 其中未被處理的原因有為 -- 上述的方式皆無效，程式已經很精簡，先直接使用以加速系統建立速度等。 最後，若要更進一步縮減的話，可以選擇直接修改程式碼，發展嵌入式的專屬版本 (如使用 BusyBox 所提供的 API)，整合更多功能的守護神程式 (使用 Apache 提供的 proxy 取代 Squid) 等。

## 6. 結論

使用 Linux 作為嵌入式作業系統，是一件非常有趣的事情，因為使用者與貢獻者遍佈世界各地，他們都貢獻出自己努力的成果，各種主要平台上都有支援的版本。 可是在資源眾多的情況下，卻沒有一個很好的整合環境，實在非常可惜。 在嵌入式 Linux 中，儘管各個獨立的開發工具，都已經功能完備又兼具圖形使用介面，實在相當不錯。 然而整合發展環境這個部份，尚在起步階段。 與嵌入式作業系統 VxWorks 的開發環境相比之下，功能就顯得太過簡單。 即使是以嵌入 Linux 為主的廠商如 Lineo[13] 的 Embedix, MontaVista[14] 的 HardHat Linux 等，都提供有不錯的開發工具，唯獨缺乏一個功能強悍的圖形化整合環境。 整合環境在功能上應以跨平台的發展環境為最重要的方向，輔以網路連線除錯功能，動態下載執行模組，即時圖形顯示執行狀態等，將會是挑戰商業軟體重要的里程碑。 我們認為這就是開放式軟體一直以來所面對的同樣問題，相信如果得到解決，嵌入式 Linux 的應用必定會快速成長。

除了在本文中所見，可以使用一般的 Linux 核心來實作以外，還可以有其他的選擇。 如不含記憶體管理單元 (MMU) 的 uclinux[15] 等，或是支援各種中央處理器的 ARMLinux[16, 17], LinuxPPC[18] 等，可根據應用的不同而作出適當的選擇。 希望你也可以使用嵌入式 Linux，應用在各個不同的領域上，令無限的創意一一實現.

## 7. 參考資料


[1] Ying-Dar Lin, Shao-Tang Yu, Huan-Yun Wei, Integrating and Benchmarking Security Gateway with Open Source Firewall, VPN, and IDS, submitted for publication, August 2001

[2] The Swiss Army Knife of Embedded Linux, http://busybox.lineo.com

[3] The worlds smallest login/passwd/getty/etc, http://tinylogin.lineo.com

[4] CrossGCC Frequently Asked Questions, http://www.objsw.com/CrossGCC

[5] Etherboot home page, http://etherboot.sourceforge.net

[6] The Redhat Embedded Debug and Bootstrap firmware, http://sources.redhat.com/redboot

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

[8] Squid Web Proxy Cache, http://www.squid-cache.org

[9] The FireWall ToolKit (FWTK) from TIS, http://www.fwtk.org

[10] Linux FreeS/WAN, http://www.freeswan.org

[11] The Open Source Network Intrusion Detection System, http://www.snort.org

[12] iproute2 + tc notes, http://snafu.freedom.org/linux2.2/iproute-notes.html

[13] Lineo, Inc., http://www.lineo.com

[14] MontaVista Software, http://www.mvista.com

[15] Embedded Linux Microcontroller Project, http://www.uclinux.org

[16] armlinux.org homepage, http://www.armlinux.org

[17] The ARM Linux Project, http://www.arm.linux.org.uk

[18] The Home of the PowerPC GNU/Linux Port, http://www.linuxppc.org