移植 NetBSD 至 ARM 嵌入式處理器

張耀中 曹世強 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

E-MAIL：{ycchang,weafon,ydlin}@cis.nctu.edu.tw

摘要

NetBSD 是一個 Open Source 的 UNIX-like 作業系統，除了有著 4.4BSD Lite 優異的血統外，穩定、模組化、跨平台是它主要的優點。至於 ARM 嵌入式處理器則是目前應用在嵌入式系統領域中的當紅炸子雞，同時也是 NetBSD 支援的處理器之一，然而要使得 NetBSD 在 ARM 嵌入式處理器上運作，勢必要經過移植 (porting) 的過程，也就是要將平台相依 (platform dependent) 的部分做適當的修改。在本文裡我們實際將 NetBSD 安裝到一個以 ARM 為處理器核心的個人數位助理 (PDA, Personal Digital Assistant) 上，以方便找出 NetBSD 和平台相依的部分。此外，透過觀察原始碼的佈局, 則可清楚的指出，開機流程和裝置驅動程式便是和平台相依有關的部分。我們會在本文中闡明其運作機制。瞭解這些部分，將是移植 NetBSD 工作的主要關鍵。

關鍵字：NetBSD、ARM、平台相依、開機流程、裝置驅動程式

1 簡介

移植的廣泛定義，是讓一套軟體可以在一套選定硬體平台上正常運作。在這篇文章中，我們談的是如何移植 NetBSD 這套作業系統到 ARM 嵌入式處理器的平台上。其中，NetBSD 是一個 Open Source 的 UNIX-like 作業系統，它萌芽自 4.3BSD Lite ，一個發展於加州柏克萊大學 (BSD, Berkeley Software Distribution) 的 Unix 版本，在往後的數年裡，許多來自 4.4BSD Lite 的修正，都已經被整合到其中[1]。除了穩定、模組化外，它最大的特色就是支援眾多的平台，截至目前為止已支援的平台超過 50 種，ARM 的處理器是其中之一。這主要歸功於核心當初在設計時和平台相依的程式碼較少;和平台不相依的程式碼較多，從表 1 中我們可以看出和平台相依的部分在 HP300 的核心原始碼裡僅佔了

將近 20%，這凸顯了它的高度可移植性。由於有開放式程式碼及高度可移植性這兩個特性，選用 NetBSD 將大幅降低廠商在開發硬體規格多變的嵌入式系統時，移植作業系統部分所需增加的負擔。

Category	Lines of code	Percentage of kernel
machine dependent headers	1562	0.8
device driver headers	3495	1.7
device driver source	17506	8.7
virtual memory	3087	1.5
Other machine dependent	10970	5.4
routines in assembly language	3014	1.5
Total machine dependent	39634	19.6

表 1. Machine-dependent part for the HP300 in the 4.4BSD kernel[2]

不過要注意的是，由於 NetBSD 配備著優異的虛擬記憶體系統，因此它只能在有記憶體管理單元 (Memory Management Unit, MMU) 的處理器上運作。

而另一方面，ARM 處理器核心是一個 32 位元的精簡指令集架構 (RISC, Reduced Instruction Set Computer)，因此，其設計較為簡單，並且耗電量低，所以被廣泛的運用到各種嵌入式系統裡。通常一顆嵌入式處理器會包含一個處理器核心還有許多的運算周邊，以我們在本文中所使用的 PDA 為例，它使用了一顆 Intel StrongARM-1110 的處理器，裡面包含了 Intel 的 StrongARM 處理器核心（相容於 ARM 的指令集架構）以及中斷控制器、計時器等等的周邊。從表 2 我們發現有些處理器核心並沒有內建記憶體管理單元。

Type	CPU Core	Cache Size (Inst/Data)	Tightly Coupled Memory	Memory Manage -ment	AHB Bus Interface	Thumb	DSP	Jazelle	Clock MHz **
Embedded Cores	ARM7TDMI	No	No	No	Yes*	Yes	No	No	133
	ARM7TDMI-S	No	No	No	Yes*	Yes	No	No	100-133
Intel ARM-based Processors	StrongARM	16K/8K	No	MMU	N/A	No	No	No	206
	Intel XScale	32K/32K	No	MMU	N/A	Yes	Yes	No	400

表 2. ARM 的處理器核心列表[3]

在這樣簡單的介紹後，很明顯的，NetBSD 與 ARM 的嵌入式處理器的組合，將是一個非常實用的嵌入式系統設計。在下一節，我們先討論移植所需考慮的事情。而第三節，我們描述了安裝 NetBSD 到一個 ARM 平台的過程。在這邊我們選用的是一台 HP 的 IPAQ 3630 個人數位助理。接著，在第四節將深入探討 NetBSD 和平台相依的部分，以瞭解要移植到 ARM 平台，所需考慮的細節。最後是我們的結論。

2 移植

什麼是移植？從軟體的角度來看，移植是使一套軟體在不同平台上正常運行的過程，在此我們的軟體是指作業系統，而平台是指硬體平台。我們都知道作業系統是介於應用程式和硬體間溝通的橋樑，因此當底層的硬體有所改變時，作業系統和平台相依 (platform dependent) 的部分就必須跟著做變動，所以在移植前，瞭解目標平台 (target platform) 的硬體規格是相當重要的前置工作。除此之外，我們對於要移植的作業系統，其中與平台相依的各個部分要先熟悉，如此才能將這些部分迅速的轉移到其他的平台上。

因此，在進行移植前，有三個問題是要先釐清的，它們關係到移植規模的大小和困難度。

- 目標平台

目標平台裡包含了所使用的處理器和處理器外的周邊裝置，而處理器中可能也整合了一些運算周邊，舉如中斷控制器、計時器等等。因此在進行移植前，我們可以從目前作業系統的支援程度來決定哪些部分要改寫以及哪些部分可以套用原本的程式碼。以移植 StrongARM-1110 為例，StrongARM-1100 和 StrongARM-1110 這兩顆處理器的核心都相同，但是後者還整合了其他的周邊，因此在新的平台上，我們可以套用初始化 StrongARM-1100 處理器核心的部分，而針對剩下的周邊撰寫新的裝置驅動程式。

- 記憶體管理單元

記憶體管理單元負責作業系統中虛擬記憶體保護的工作，NetBSD 虛擬記憶體的機制就是靠它來完成的，當我們要將 NetBSD 移植到一個沒有記憶體管理單元的平台上，所需要做的修改就非常的多了，因此在選擇目標平台的處理器時要相當的謹慎。

- 記憶體對映 (memory map)

嵌入式系統，通常為一個無磁碟的裝置，並且配置了有限容量的 SDRAM 和 FLASH ROM，其中記憶體控制器 (memory controller) 負責兩者在處理器位址空間 (address space) 的對映，不同於 PC 的是，每種平台的記憶體對映位置不一定相同，因此會有平台硬體規格相同但是記憶體對映位置不同的問題產生，所以在進行移植時可能只需要藉由重新規劃記憶體控制器就可以使某平台的核心在新平台上面運作。

3 安裝 NetBSD 至 ARM 處理器平台

在前一節我們瞭解，要將 NetBSD 移植到 ARM 嵌入式處理器上，必須先釐清楚 NetBSD 哪些部分為平台相依的，這也正是本文要探討的問題。但在探討相依問題前，我們先試著將 NetBSD 安裝在以 ARM 為處理器核心的個人數位助理 (PDA, Personal Digital Assistant) 上。因為，我們認為，要瞭解一個系統，將它實際安裝並且執行是瞭解他們相依性的第一步驟。

3.1 硬體描述

我們要安裝的目標 (target) 是一台 HP 的 IPAQ 3630 個人數位助理，它是嵌入式系統的一個典型範例，裡面使用了一顆 Intel 的 StrongARM-1110 嵌入式處理器。從圖 1 中我們可以看出這顆處理器包含了 StrongARM 的處理器核心以及許多的周邊，比較特別的是內建了一顆 LCD 控制器，歸納一下表 2 以及規格文件

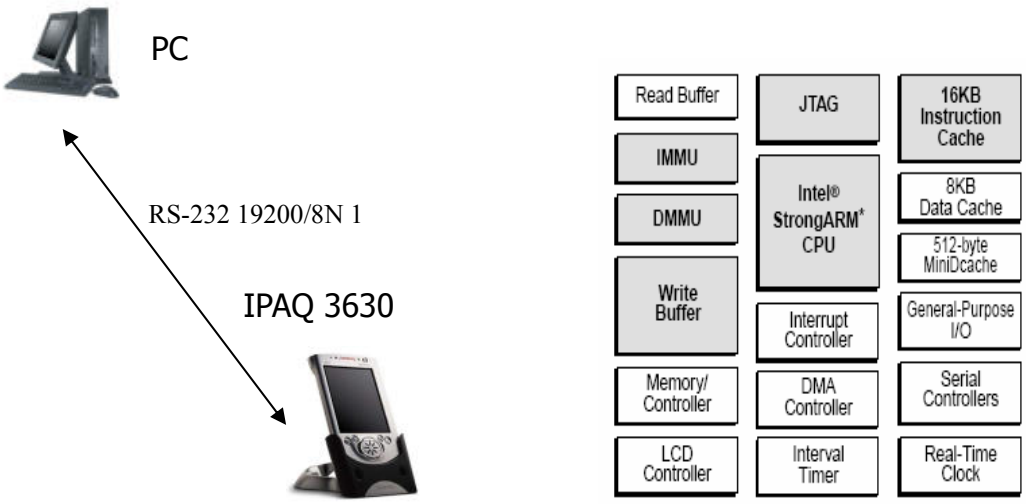

圖 1. Intel StrongARM-1110 嵌入式處理器功能方塊圖[4]

上[4]的資料，我們得到如表 3 的結果。同時，我們還必須準備一張 16MB 的 CF 卡以及專用背夾做為存放開機程式 (bootstrap loader) 和核心影像檔 (kernel image) 的儲存裝置。此外，我們還必須要準備一台 PC，上面存放著 NetBSD 的原始程式碼，稍後我們會在此機器上編譯給目標平台執行的核心影像檔，並且使用 RS-232 連接線來傳輸。

處理器核心	Intel StrongARM 206Mhz
SDRAM	32MB
FLASH ROM	16MB
內建 MMU	YES
SDRAM 起始位置	0x40000000
NetBSD 是否支援	YES

表 3. HP IPAQ 3630 重要硬體規格

3.2 進行方式

取得最新的原始碼

首先，我們必須在 PC 端取得 NetBSD 最新的原始程式碼，截至目前為止，最新的版本是 1.6.1 RELEASE。

```
STEP1. # cd /usr/src
STEP2. # ncftp ftp://netbsd.csie.nctu.edu.tw/pub/NetBSD/NetBSD-1.6.1/source/sets/
STEP3. 分別取得 gnusrc.tgz、src.tgz、syssrc.tgz、sharesrc.tgz
STEP4. 將這些 tarball 個別解開到/usr/src 下
```

編譯跨平台工具鏈 (cross tool chains)

在幫目標平台編譯好核心影像檔之前，我們必須要先準備一個完備的跨平台編譯 (cross compile) 環境，之所以這樣做的原因是因為 PC 端原本所使用工具鏈（包括組譯器、編譯器等等工具）所產生出來的是 x86 的機械碼，因此我們必須要打造一個專門產生 ARM 機械碼的工具鏈，如此編譯出來的核心影像檔才能在目標平台上正常執行。

```
STEP1. # cd /usr/src
STEP2. # ./build.sh –m hpcarm –T /usr/cross tools 其中 /usr/cross 放著稍後編譯好的跨平台工具鏈
```

編譯核心影像檔

有了跨平台工具鏈之後，我們就可以開始進行編譯核心影像檔的工作，其步驟如下：

```
STEP1. # cd /usr/src/sys/arch/hpcarm/conf
STEP2. # /usr/cross/bin/nbconfig IPAQ 其中 IPAQ 是核心設定檔
STEP3. # cd ../compile/IPAQ
STEP4.# /usr/cross/bin/nbmake-hpcarm depend all
STEP5.# gzip –c netbsd > netbsd.gz
```

經過數分鐘的編譯後，在編譯目錄下會產生一個 ELF 格式的 netbsd 核心影像檔。然而，我們必須將它轉換成一個 gzip 的壓縮格式，因為開機程式只認得這樣子的檔案格式。

安裝開機程式和核心影像檔

接著，我們必須將開機程式和核心影像檔傳輸到目標平台上，也就是 IPAQ 3630。特別注意的是，開機程式是一支 WINCE 的應用程式，它主要的運作方式是將 NetBSD 核心影像檔解壓並且放置在 0x40000000 的記憶體位置，同時將控制權交給核心的進入點，因此我們需要將這兩支程式透過 RS-232 或是讀卡機的方式傳輸到背夾的 CF 卡上。

，

STEP1. # cd /usr/src/sys/arch/hpc/stand/binary/ARM

STEP2.# uudecode -p hpcboot.exe.uu >hpcboot.exe

STEP3. 將 netbsd.gz 和 hpcboot.exe 分別傳輸到背夾的 CF 卡上

開機

最後，我們開始進行開機的程序，一開始我們先在 IPAQ 3630 上點選 hpcboot 這支開機程式，設定如圖 2 所示。

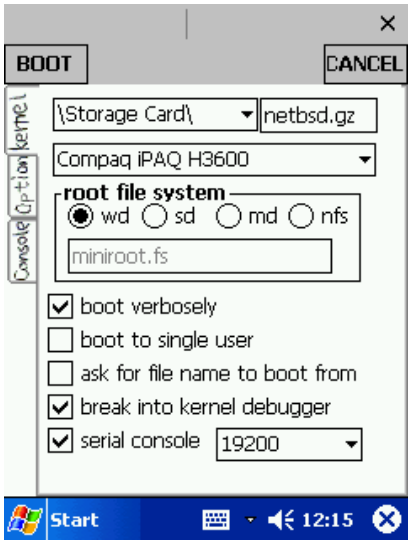

圖 2. 開機程式的設定畫面

其中，serial console 的設定尤其重要，因為目前 NetBSD 在 LCD 控制器上的處理還有些問題，因此所有的輸出都得透過 serial console 來表示。另外在 PC 端的部分，我們須執行 minicom 或是 tip 等終端機程式，並且將序列阜設定為：19200 bps、8 位元傳輸、沒有同位元檢查。待一切都沒問題後，按下開機程式的 BOOT 鍵，我們就可以在 PC 端的終端機上看到如圖 3 的開機畫面。

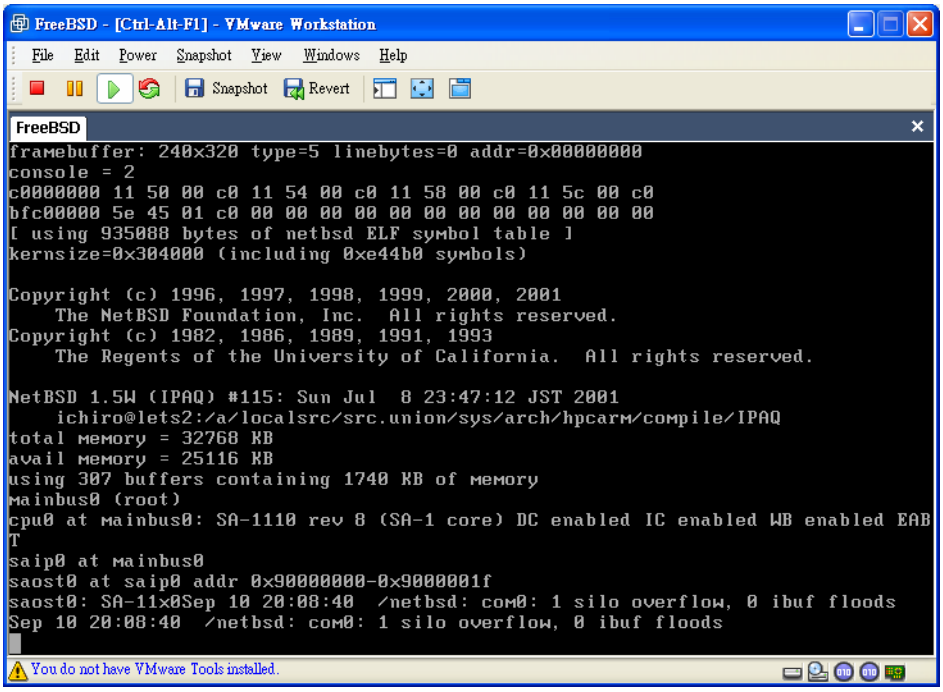

圖 3. 開機成功的終端機畫面

4 深入 NetBSD 和平台相依的部分

現在我們已經能夠在 ARM 處理器平台上運行 NetBSD 了，接著我們回到本文所要探討的問題，NetBSD 和平台相依的部分。首先，我們先從核心原始碼的佈局來描繪和平台相依的地方。表 4 主要說明了核心原始碼中各個目錄所代表的意義，其中在/usr/src/sys/arch 的目錄底下包含了許多平台的子目錄，而主要跟 IPAQ 3630 有關的目錄為/usr/src/sys/arch/hpcarm 以及/usr/src/sys/arch/arm。然而，從這些目錄裡的程式碼所做的事情當中，我們歸納出兩種主要的動作：開機環境的建置以及周邊裝置的驅動，因此我們以下針對開機的流程 (bootstrap) 和裝置驅動程式這兩個和平台相依的部分，來深入探討它們的運作機制。

層次	意義
adosfs, coda, filecorefs, isofs, msdosfs, nfs, ntfs, ufs/*,miscfs/*	各種檔案系統的原始碼
compat/*	一些相容於舊版本的核心函式庫
net*/	和網路有關的核心原始碼
stand	一些工具程式
lkm	核心可掛載的模組
sys	核心原始碼所使用到的標頭檔

uvm	虛擬記憶體系統原始碼
arch	各種和平台相依的核心原始碼
dev/*	跨平台的裝置驅動程式
ddb	核心除錯器

表 4. /usr/src/sys 下比較重要的目錄所代表的意義[5]

4.1 開機的流程

從圖 4 中我們可以知道 NetBSD 的核心影像檔主要是透過開機程式 (bootstrap loader)，在此指的是 hpcboot，將它載入到特定的記憶體位址，這個位址通常是 SDRAM 的起始點，由內部的記憶體對映 (memory map) 來決定。載入後，此時程式的控制權會由開機程式交還給核心的進入點，也就是一些組合語言指令，藉由這些組合語言指令來初始化 C 語言執行的環境，接著才能跳躍到 initarm()以及 main()這兩個用 C 寫的函式。

其中，initarm()這個函式主要的工作就是在建置核心開機的環境。何謂核心開機的環境呢？一開始我們曾經提到，NetBSD 是一個在虛擬記憶體模式下運作的作業系統，在虛擬記憶體機制還沒啟動前，我們必須先初始化處理器內部負責管理虛擬記憶體的記憶體管理單元 (Memory Management Unit, MMU) ，如此才能在啟動分頁 (paging) 前，把一些重要的資料結構給配置好，舉如 Level 1、Level 2 的分頁表 (page table) 。從圖 5 中我們可以看到，在初始化記憶體管理單元的過程中，使用到一些以 pmap 開頭的函式名稱，這些函式統稱為 pmap module，裡頭定義了一些操作記憶體管理單元的方法。

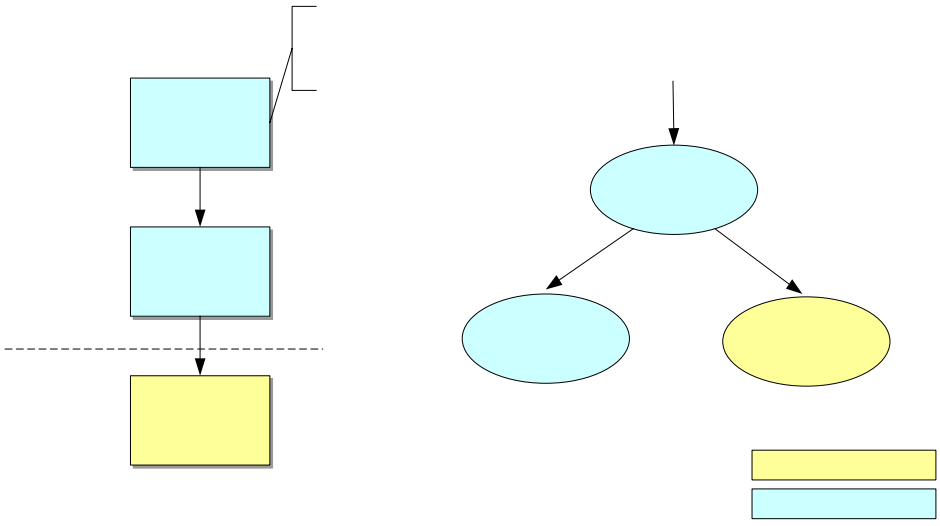

圖 4.NetBSD 在 IPAQ 3630 開機的主要步驟

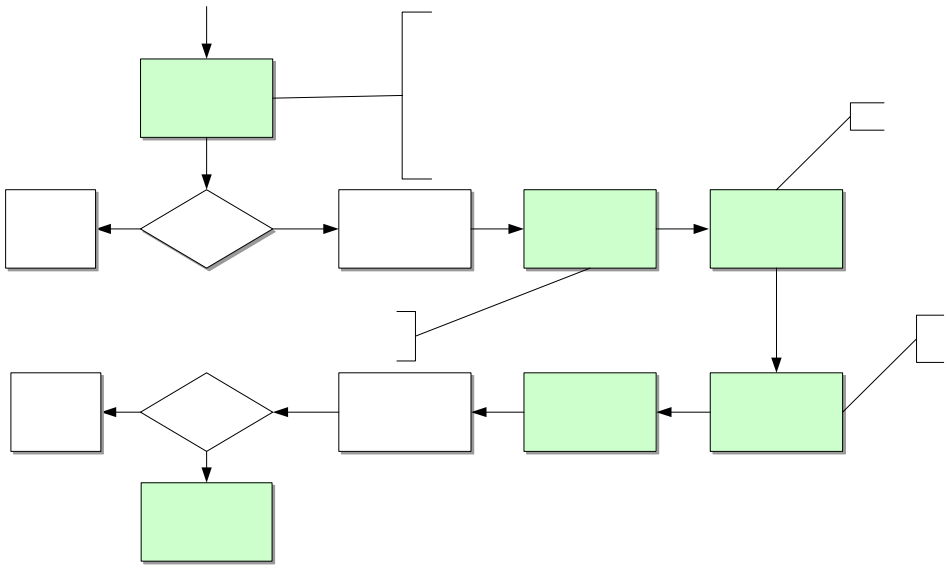

圖 5.initarm()- /usr/src/sys/arch/hpcarm/hpcarm/hpc_machdep.c

我們透過圖 6 來說明 pmap module 與核心上層之間的關係，基本上，pmap module 主要提供給核心上層，那些需要修改實體記憶體 (physical memory) 對映的系統，一組相同的介面。NetBSD 所使用的 UVM 就是一個例子，它是整個系統中負責處理虛擬記憶體空間配置的子系統，當對映實體記憶體空間的要求產生時，好比說 Swap in，UVM 就會利用 pmap_enter()這個介面來做虛擬與實體記憶體之間的對映，這也就是為什麼 UVM 不是平台相依的主要原因了。但是我們都知道，每種處理器的記憶體管理單元，它們的操作方式不一定都相同，因此當我們要將 NetBSD 移植到新的處理器時，就必須考慮到 pmap module 的平台相依性。

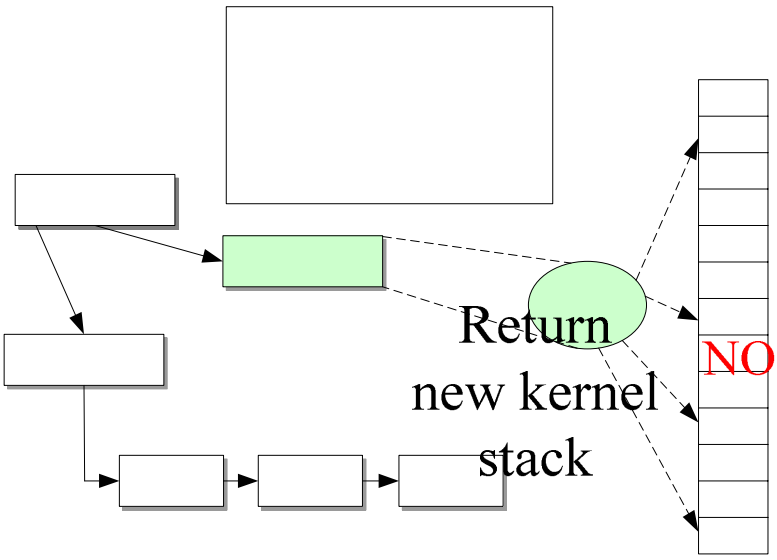

圖 6. pmap module 和 UVM 之間的關係圖

另外，圖 5 中還說明了一件事情，在進入點的一開始，先呼叫 set_cpufuncs() 的函式，它主要的任務就是在判斷這顆正在運作的處理器是否有支援。它的判別方法主要是這樣子，首先，先藉由處理器中協同處理器 (coprocessor) 指令的幫助，可以取得一組 CPU 的識別碼，根據判別這個識別碼，核心可以知道是否有包含操作該處理器的一組介面在裡面，如果有的話，舉如圖中的 arm9_cpufuns，之後只要跟 CPU 有關的操作，像是啟動 cache、清除 TLB 等等，都會使用到這組介面裡所定義的函式，反之，如果沒有，核心就會進入當機模式。因此，如果你的處理器沒有支援的話，就必須自己實作這組介面。

在建置好核心開機的環境之後，接著會進入 main()這個重要的函式，基本上我們可以把它當成是核心正式的一個進入點，在這個進入點裡，核心其他的子系統都會跟著被初始化。其中裝置驅動程式初始化的起始點就是在 configure()這個函式裡，如圖 7.所示，因此接下來我們就來介紹裝置驅動程式初始化的運作機制。

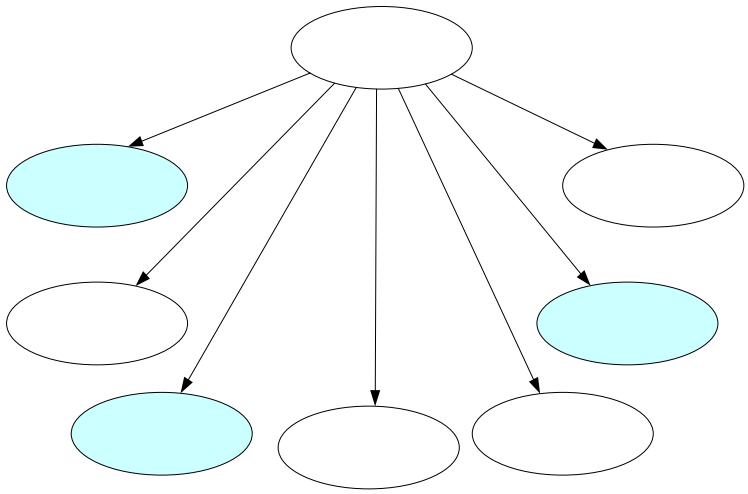

圖 7. main()主要呼叫的子系統

4.2 裝置驅動程式

在 NetBSD 裡，所有的裝置驅動程式均遵守自動組態設定 (autoconfiguration) 這個架構所訂定的規則，它主要的動作就是在初始化包含於核心裡的裝置驅動程式。每個裝置驅動程式基本上必須提供一個介面，雛形如表 5 所示，告知自動組態設定系統，當要掛載 (attach) 它時需要呼叫哪一個函式、要配置給它多少的記憶體空間大小等等的動作。接著，必須透過 config 這個工具建置一些給自動組態

```
struct cfattach foo_ca = {
        sizeof(struct foo_softc),          /* size of instance data */
        foo_match,                         /* match/probe function */
        foo_attach,                        /* attach function */
        foo_detach,                        /* detach function */
        foo_activate                       /* activate function */
};
```

表 5 用於裝置驅動程式的 cfattach 結構[6].

設定系統看得懂的表格，如圖 7 所示，而這些表格會在編譯時包含在核心影像檔裡，在 configure()函式被觸發後，利用它們來進行裝置驅動程式初始化的動作。

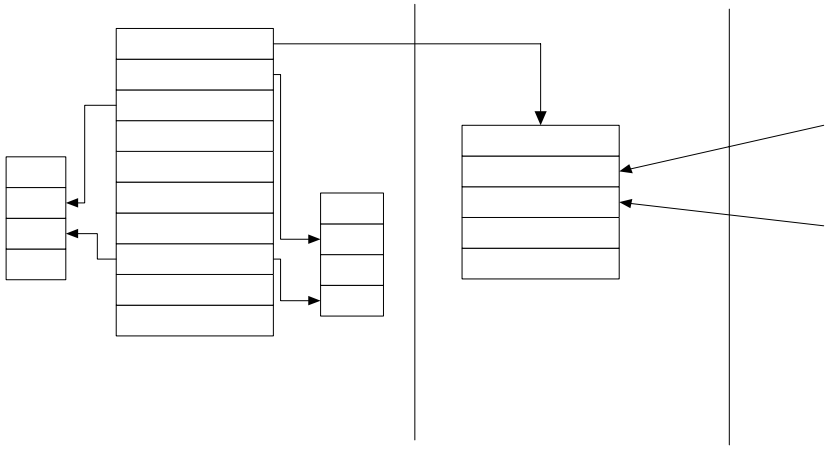

圖 7. /usr/src/sys/arch/hpcarm/compile/ioconf.c

然而，經過我們的觀察，發現這些表格主要在建立一個如圖 8.的樹狀結構，簡稱為裝置樹 (device tree)，透過樹的尋訪，以 depth first search (DFS) 的方式，從樹根依序呼叫所尋訪節點的 match()函式來決定是否要將它載入。其中有些節點為虛擬的裝置，舉如 cpu0，而有些則為匯流排裝置，舉如 mainbus0、saip0、ipaqbus0、pcmcia0。在匯流排裝置的驅動程式裡，一些介面對於它的子節點來說是相當重要的，我們以圖 8.中 ipaqcic0 和 pcmcia0 這兩個裝置的關係來做說明。

pcmcia0 的裝置驅動程式基本上是位於/usr/src/sys/dev 底下，也就是說它是一個和平台無關的裝置驅動程式，但是我們可能會問，那註冊中斷機制等等和平台相依的介面是誰在提供的呢？答案就在它的父節點，ipaqcic0，換言之，當我們在移植不同的平台時只要父節點提供給 pcmcia0 相同的使用介面，那麼就可以達到 pcmcia0 和平台無關的特性，PCI 裝置也是如此。

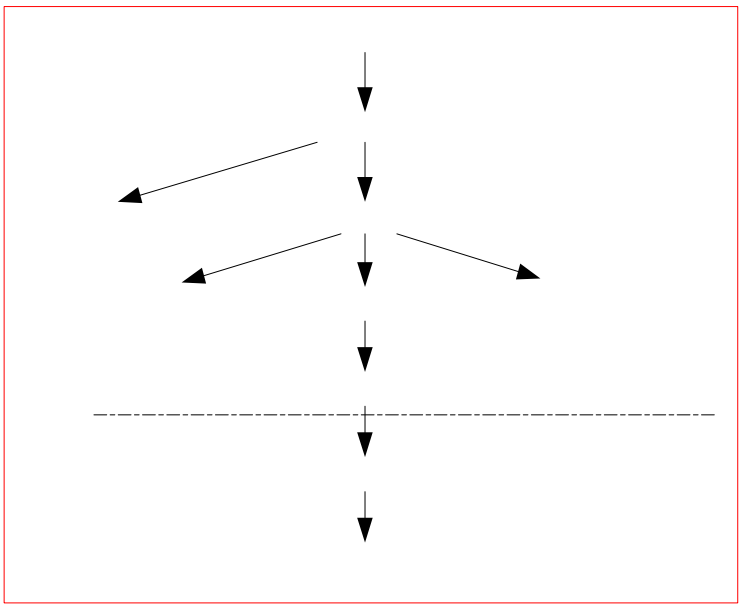

圖 8. device tree traversal

5 結論

移植是一個蠻有趣的過程，除了可以瞭解作業系統各個與平台相依的部分外，更可以知道支配作業系統運作的硬體，它們主要的操作方式。在這份報告中，我們嘗試安裝 NetBSD 到 ARM 平台，以方便我們進一步瞭解平台移植時所會遇到的問題。根據我們實驗及觀察程式碼的結果們發現要移植平台，必須改變的為與平台相依的兩個動作：開機環境的建置以及周邊裝置的驅動。前者，必須根據平台的記憶體對映，建置與開機環境有關的分頁表，其中，pmap module 扮演著操作記憶體管理單元的角色，因此我們需要針對不同的記憶體管理單元來修改 pmap module 底層的操作方法。後者，藉由 NetBSD 裝置樹獨特的概念，從樹的走訪中優雅的掛載裝置驅動程式，同時，透過樹的分層，父節點這個匯流排裝置驅動程式，只要提供給子節點相同的介面，就算使用不同廠商的匯流排控制晶

片，如 PCI，也能使得子節點裝置驅動程式，如乙太網路卡，具有和平台無關的特性。整個移植過程裡，最困難的地方在於是否熟悉硬體架構，因為許多和平台相依的部分都是針對硬體本身的特性和運作方式來設計，因此若想要踏入移植這個領域，勢必要在瞭解硬體的特性和運作方式上下功夫。

6 參考資料

[1] http://www.netbsd.org/Misc/about.html, *The History of the NetBSD Project*.

[2] McKusick, Bostic, Karels, "*Quarterman, The Design and Implementation of the 4.4BSD Operating System*, " Addison Wesley, 1996.

[2] http://www.netbsd.org/Misc/about.html, *The History of the NetBSD Project*.

[3] http://www.arm.com/armtech/core_selection?OpenDocument, *Core Selection Guide.*

[4] http://www.intel.com/design/strong/manuals/278240.htm/, *Intel® SA-1110 Microprocessor Developer's Manual.*

[5] http://ezine.daemonnews.org/200205/netbsdsrctree3.html, *A Tour through the NetBSD Source Tree - Part III: Kernel.*

[6] "*driver - description of a device driver,"* NetBSD Kernel Manual.