



國立臺北科技大學

資訊工程系碩士班

碩士學位論文

Android 作業系統移植之研究與實現
The Study and Implementation of Operating
System Porting for Android

研究生：鍾文昌

指導教授：梁文耀

中華民國九十八年七月

摘要

論文名稱：Android 作業系統移植之研究與實現

頁數：132

校所別：國立臺北科技大學資訊工程系碩士班

畢業時間：九十七學年度第二學期

學位：碩士

研究生：鍾文昌

指導教授：梁文耀 博士

關鍵詞：Android、ARM、Cell、Device driver、Embedded、GNU Make、Google、Kernel、Linux、Mobile、Open Source、Operating System、Phone、Porting、PXA270、手機、作業系統、移植、嵌入式

近年來，智慧型手機崛起，伴隨著手機作業系統也逐漸受到重視，以往手機系統大多是各手機大廠自行研發且封閉的系統；但自從 Google 開始覬覦手機市場的廣告商機，而著手開發一個完全免費且開放式的手機軟體平台 Android，全球各相關產官學界便開始爭相研究，也替 Linux 在智慧型手機市場注入一劑強心針。

一般有能力移植 Android 的業界不願公開移植的步驟及流程，而學界則恐缺資源及人才，因此不易將 Android 移植至實際平台，我們不僅成功地將 Android 移植至 PXA270，並公開移植的步驟、流程、所撰寫及修改的程式碼，使他人可以仿照此流程，建立 Android 的執行環境。

本論文的重點在於移植的流程，而非硬體平台，此流程適用於多數的硬體平台，但不可能百分之百適用於所有的開發環境。

我們為此打造一個全新的開發環境，使其在 GNU Make 支援下盡可能執行 parallel make 以縮短編譯時間。

我們希望透過本論文能夠一窺 Android 的運作方式，並分享移植過程中所遇到的問題及解決方法。

ABSTRACT

Title : The Study and Implementation of Operating System Porting for Android

Pages : 132

School : National Taipei University of Technology

Department : Department of Computer Science and Information Engineer

Time : July, 2009

Degree : Master

Researcher : Wen-Chang Chung

Advisor : Wen-Yew Liang

Keywords : Android, ARM, Cell, Device driver, Embedded, GNU Make, Google, Kernel, Linux, Mobile, Open Source, Operating System, Phone, Porting, PXA270

In recent years, with the appearance of smart phones, development of the operating systems for handheld devices has attracted more and more attention. In the past, the specifications and developments of the handheld devices are usually defined by the vendors in a proprietary method. With the ambition to extend the market share in the mobile phone related industry, Google has started to develop a free and open source based software platform, called Android, for these kinds of devices. Since then, Android soon gets focus from the world-wide industry companies, government departments, and academic organization. This also helps the Linux community in its applications in embedded systems.

We not only have ported Android into one of our embedded platform (a PXA270 board), but also we'd like to share our experience and detailed procedures to anyone who are interested in Android porting. The keypoint of this thesis is to emphasize the process of porting, instead of how to port it to any particular hardware platform. However, this process still depends on some of the hardware components that are used. The porting process involves many issues related to the hardware. These include the work of kernel porting, driver porting, and also some modification to the Android

application framework and libraries.

Based on our work, we hope that people who are interested in Android have the chance to get more understanding on how it work and how to realize it on their own platforms.



誌 謝

首先要感謝指導教授梁文耀老師悉心教誨並指示研究方向，促使本論文能夠順利完成。

感謝口試委員方志鵬老師、張陽郎老師、蘇慶龍老師以及指導教授梁文耀老師百忙之餘，撥冗擔任學生的口試委員，感謝各位老師的悉心指教，並提出諸多寶貴建議，使本論文更加完善。

謝謝家人、朋友及實驗室各位夥伴的幫忙及協助，讓我可以順利完成碩士學業。

最後特別感謝正浩幫我拍攝論文中所需的照片。



目 錄

中文摘要	i
英文摘要	ii
誌 謝	iv
目 錄	v
表目錄	ix
圖目錄	x
第一章 緒論	1
1.1 前言	1
1.2 研究動機與目的	1
1.3 研究貢獻	2
1.4 論文架構	3
第二章 Android 簡介	4
2.1 硬體限制	4
2.2 軟體特色	5
2.3 開發環境	8
2.4 Linux 手機聯盟	8
第三章 一般 Embedded Linux System 簡介	10
3.1 系統架構	10
3.2 開機流程	10
3.2.1 在 ARM 架構下的開機流程	11
3.3 開發流程	17
第四章 Android 移植步驟及流程	19
4.1 實驗平台 PXA270 的硬體概況	19

4.2	準備工作	22
4.3	移植 Linux kernel.....	24
4.3.1	使用工具	26
4.3.2	可能遇到的問題及解決方法	28
4.3.3	編譯 Linux kernel.....	30
4.3.4	測試 Linux kernel.....	30
4.4	整合性修改	30
4.4.1	LCD	32
4.4.1.1	Double Framebuffer.....	32
4.4.1.2	LCD 畫面閃爍	33
4.4.2	Keypad.....	33
4.4.3	Touch	34
4.4.4	編譯 Android.....	40
4.4.5	init.rc	40
4.5	所有修改及新增的程式碼列表	41
4.5.1	Linux kernel / device driver.....	41
4.5.2	Android	43
4.6	執行 Android.....	44
4.7	移植成果	45
第五章	開發環境簡介	48
5.1	一般傳統 Building Environment 與 Android Building Environment 簡介	48
5.2	我們的開發環境簡介	50
5.2.1	版本控制系統	52
5.2.2	如何使用我們的開發環境	54
5.2.3	測試平台	59

第六章 結論	61
參考文獻	62
附錄 A Shell 簡介	65
附錄 B Framebuffer driver (pxafb.c)	66
附錄 C pxafb.h	69
附錄 D Keypad driver (android_keypad.c)	70
附錄 E Touch driver (ucb1400_ts.c)	80
附錄 F init.rc	85
附錄 G linux-2.6.25-android-1.0_r1/Makefile	91
附錄 H linux-2.6.25-android-1.0_r1/arch/arm/Makefile	92
附錄 I linux-2.6.25-android-1.0_r1/arch/arm/configs/android_pxa270_defconfig	93
附錄 J linux-2.6.25-android-1.0_r1/arch/arm/kernel/head.S	103
附錄 K linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/clock.c.....	104
附錄 L linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/irq.c.....	105
附錄 M linux-2.6.25-android-1.0_r1/drivers/cpufreq/Kconfig	106
附錄 N linux-2.6.25-android-1.0_r1/drivers/i2c/chips/Makefile	107
附錄 O linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Kconfig	108
附錄 P linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Makefile	109
附錄 Q linux-2.6.25-android-1.0_r1/include/linux/config.h	110
附錄 R mydroid/cdma-import/build/core/definitions.mk	111
附錄 S mydroid/cdma-import/external/sqlite/dist/Android.mk.....	112
附錄 T mydroid/cdma-import/frameworks/base/core/jni/server/com_android_ser ver_BatteryService.cpp	113
附錄 U mydroid/cdma-import/hardware/libhardware/power/power.c	115

附錄 V 建立 Android 執行環境的操作步驟	116
附錄 W Android 相關的參考資料	117
作者簡介	119



表目錄

表 4-1	實驗平台 PXA270 硬體規格	21
表 5-1	我們的開發環境在不同平台上的測試數據	60



圖目錄

圖 2-1	Android Architecture	6
圖 2-2	Android Emulator	7
圖 3-1	一般 Embedded Linux System 架構	10
圖 3-2	一般 Embedded Linux System 開機流程	11
圖 3-3	Bootloader 與 Linux kernel 的 architecture number 比對失敗	11
圖 3-4	選擇 Kernel hacking	12
圖 3-5	開啟 CONFIG_DEBUG_LL 選項	13
圖 3-6	因為 bootloader 與 Linux kernel 的 architecture number 不同，在開啟 Linux kernel config 中的 CONFIG_DEBUG_LL 後所顯示的除錯訊息	14
圖 3-7	Linux kernel 開機流程	15
圖 3-8	init 執行流程	16
圖 3-9	一般 Embedded Linux System 的目錄結構	17
圖 3-10	一般 Embedded Linux System 的建構流程	17
圖 3-11	我們的開發環境	18
圖 4-1	PXA27x Processor Block Diagram	20
圖 4-2	實驗平台 PXA270	21
圖 4-3	Android 移植步驟	24
圖 4-4	移植 Linux kernel 的流程圖	25
圖 4-5	WinMerge	27
圖 4-6	Meld	27
圖 4-7	LCD 畫面閃爍，由左至右每張圖像拍攝的時間間隔為 1 秒	33
圖 4-8	Android 對 touch 的反應	37
圖 4-10	Android 在實驗平台 PXA270 上執行時所顯示的訊息	37

圖 4-10	在 Android 環境下，輸入裝置與電源管理的關係.....	38
圖 4-11	Android 的 BatteryService 流程	39
圖 4-12	Android 的 PowerManagerService 流程	40
圖 4-13	Android 在我們實驗平台 PXA270 上的目錄結構.....	45
圖 4-14	Android 在我們實驗平台 PXA270 的操作圖像.....	46
圖 4-15	Android 在我們實驗平台 PXA270 的操作影片	47
圖 5-1	傳統 recursive make 與 Android 開發環境的比較	50
圖 5-2	我們的 building environment.....	52
圖 5-3	我們開發環境的目錄結構	54
圖 5-4	make menuconfig	55
圖 5-5	選擇 toolchain 版本	56
圖 5-6	選擇 Busybox 版本.....	57
圖 5-7	選擇 Linux kernel 版本.....	58
圖 5-8	選擇檔案系統	59

第一章 緒論¹

1.1 前言

在撰寫本論文的同時，我們一直反覆思索，要如何用淺顯易懂的方式描述移植過程，使得本論文可以被簡單地閱讀。我們深刻感受到使用文字詳述移植過程遠比實際移植來的困難，所以如果您正在參考本論文，請不要因為內容乍看過於複雜而就此卻步，因為實際移植遠比想像中的容易許多。我們不希望本論文的定位僅是Linux[1]的進階者；我們希望能讓想接觸Linux，或是Linux的新手也能夠了解Android[2]的移植流程。所以本論文以描述移植流程的方式進行，因為唯有了解流程才能夠對不同的平台進行移植工作。這也是本論文盡量不提及硬體製造商資訊的原因，因為本論文介紹的方法不限於任何硬體平台，但也不可能百分之百適用於所有的開發環境。

台語有一句話：江湖一點訣，講破無價值。因為 Android 的議題非常新，所以移植 Android 的困難在於不知道如何開始，而且在不同的平台可能會遇到不同問題；但實際上移植 Android 只是一件苦工。希望本論文能夠對於 Android 移植流程提供微薄的貢獻。

1.2 研究動機與目的

自從Google覬覦手機廣告的龐大商機，大張旗鼓宣佈正式進軍手機市場，相關新聞就開始炒得沸沸揚揚。Google為此打造一個免費、開放且高度整合的手機軟體平台，企圖借此軟體平台正式進軍手機產業，並於西元 2008 年第四季正式推出第一款Google Android手機T-Mobile G1[3] / HTC Dream[4]，且正式公開Android

¹ 在此聲明：與 Android 相關的所有開發過程，包含開發環境的建置、程式碼的撰寫及修改皆由我一人獨立完成，並無從硬體製造商得到任何支援，所開發的成果也與硬體製造商毫無關聯。強烈反對硬體製造商利用本論文從事任何相關的商業行為。

軟體平台的程式碼，供全球手機相關製造商及任何獨立開發工作者免費使用，眾多業界及自由軟體工作者也紛紛試圖將Android移植至各種不同的平台上，目前可以看到相關的研究成果[5][6]。

縱使Android已經成為當下最熱門的話題之一，學術界大多還是以模擬器或是直接購買市面上所販售的Android手機為實驗平台，原因不外乎缺乏硬體資源、軟體人才，而且Android原始碼才公開不久，因此較少硬體廠商提供相關BSP（Board Support Package，一般硬體廠商會提供所販售硬體的相關軟體套件以方便客戶開發使用，此軟體套件即稱為BSP）的支援。因為移植工作需要相當多的相關知識，包含對Linux kernel、Linux device model、Linux device driver、Linux開機流程、Linux操作環境及相關開發工具有一定程度的認識及了解，以學術單位的人力資源及經驗，比較不容易獨立將Android移植至實際平台上，再加上Android原始碼才公開不久，若是在移植上遇到問題，比較難找到對應的解決方法。我們在沒有任何硬體廠商支援的條件下，不僅將Android移植至實際平台PXA270[7]，更從無到有打造一個可供研究及開發的實際應用平台，讓我們可以從中一窺Linux kernel、Linux device driver及Android的運作方式，並分享移植過程中所遇到的問題及解決方法。

1.3 研究貢獻

1. 移植 Android 至實際平台 PXA270，證明 Android 可以在 PXA270 執行。
2. 公開移植的詳細步驟及流程，使他人可以仿照此流程，重建 Android 在 PXA270，甚至是其它平台的執行環境。
3. 撰寫並修改 PXA270 所缺少的部分 Linux device driver，使得 LCD、Keypad、Touch、USB、有線網路可以在 Android 環境下正常運作。
4. 一般有能力移植 Android 的業界不願公開移植的步驟，而學術界想要移植卻通常沒有能力，在避免對硬體製造商侵權的條件下，我們公開移植過程中所修改的相關原始碼。

5. 我們為整個開發環境打造一個全新的building environment，此building environment部分參考自The Android Open Source Project[8]，避免recursive make所造成的問題，並在GNU[9] Make[10]可以正常執行parallel make的情況下，比傳統recursive make及一般make有更高的performance。

1.4 論文架構

本論文分六大章，章節安排如下：第一章說明研究動機及貢獻。第二章為Android簡介。第三章介紹一般Embedded Linux System。第四章詳述Android移植步驟及流程。第五章介紹開發環境。最後第六章則是結論。



第二章 Android簡介

Android為Google踏入手機產業的試金石，其初步發展方向為提供一個開放式的軟體平台，讓全世界的Developer及使用者可以自行在此平台上開發或安裝相關軟體，期望此平台在短時間內受到全世界的注目及歡迎。Android的發展重點為Library及Application，因為唯有透過簡單易用的UI（User Interface）及多樣化的軟體附加功能才能夠讓Google迅速進入手機產業並佔有一席之地，這也是Google提供Android Developer Challenge²一千萬美金作為獎金的目的。

在撰寫本論文的同時，我們發現部份 Android 的官方網址有所變動，經過我們再次查證，部份相關網址已經不存在，這也表示 Android 正以飛快的速度前進，因此本論文所提供的參考網址及相關資料均為西元 2009 年 6 月 23 日之前所摘錄。Google 在此時已經推出 Android 1.5；但本論文的重點在於移植的流程，而非版本上的差異，所以本論文以 Android 1.0 為研究及移植的主軸。

2.1 硬體限制

Google官方建議執行Android的最低硬體需求³（在此指target端⁴）：

1. CPU：ARM-based
2. RAM：128MB
3. Flash：256MB

我們建議執行 Android 的最低硬體需求：

1. CPU：ARM9 Family 以上，且須支援ARMv5 以上指令集。Android Emulator[11]使用的CPU代號為Goldfish，是ARM926EJ-S，ARMv5TEJ。

² 參考網址 <http://code.google.com/android/adc.html> 已被移除。

³ 可參考 Android 原始碼中 [development/pdk/docs/system_requirements.html](http://code.google.com/android/development/pdk/docs/system_requirements.html)，如何下載 Android 原始碼可參考 4.4。

⁴ 可參考 3.3。

2. RAM：128MB 以上。我們實際移植的平台僅有 64MB，因為記憶體容量太小而造成 Android 部份應用程式在開啟時須等待數秒鐘，且 128MB 是一般智慧型手機的基本配置。
3. Flash：128MB 以上。一個基本的 Android 執行環境約需 50~100MB，而我們實際移植的平台僅有 32MB，所以我們將 Android 的檔案系統置於 USB 隨身碟中。

2.2 軟體特色

Android 的軟體架構採用分層設計的概念，此架構的優點是減少各層之間的相依性、便於獨立開發、容易收斂問題及除錯等等。

Android Architecture如圖 2-1：

1. 紅色的部份為作業系統。因為智慧型手機的功能及周邊裝置日趨繁雜，包含 Camera、Bluetooth、WiFi、GPS、LCD、Touch、Battery 及特別為手機或行動裝置所設計的電源管理系統，所以我們需要一個 OS（Operating System）來管理所有的周邊及硬體裝置，避免硬體資源被不當使用而產生不可預期的後果。Android 採用 Linux 為其作業系統，目前支援 2.6 以上版本，Android 正式公開的第一版原始碼：Android 1.0，使用 Linux kernel 2.6.25。
2. 綠色的部份為 Library。Android 採用大量的 open source software 作為其主要 Library，這也是 Android 的優點之一，因為許多 open source software 都經過一定時間的驗證，其穩定度及效能也都有一定的水準，這也是為什麼 Android 可以快速發展的原因之一，因為它站在巨人的肩膀上。
3. 黃褐色的部份主要為 Virtual Machine。Google 開發一個支援 Java 語法的 Virtual Machine：Dalvik，其特色是 Register-based Virtual Machine，而非 Stack-based Virtual Machine（Java Virtual Machine），因此 Dalvik 可以針對

其支援的平台作最佳化處理，這也是針對嵌入式系統所作的設計。

4. 藍色的部份為 Application Framework 及 Applications。Google 為此打造一系列的 Framework，希望開發人員能夠藉此加快開發速度，這也是 Android 在軟體架構上採用分層設計的考量，此架構有便於獨立開發、易於維護及容易除錯等優點。

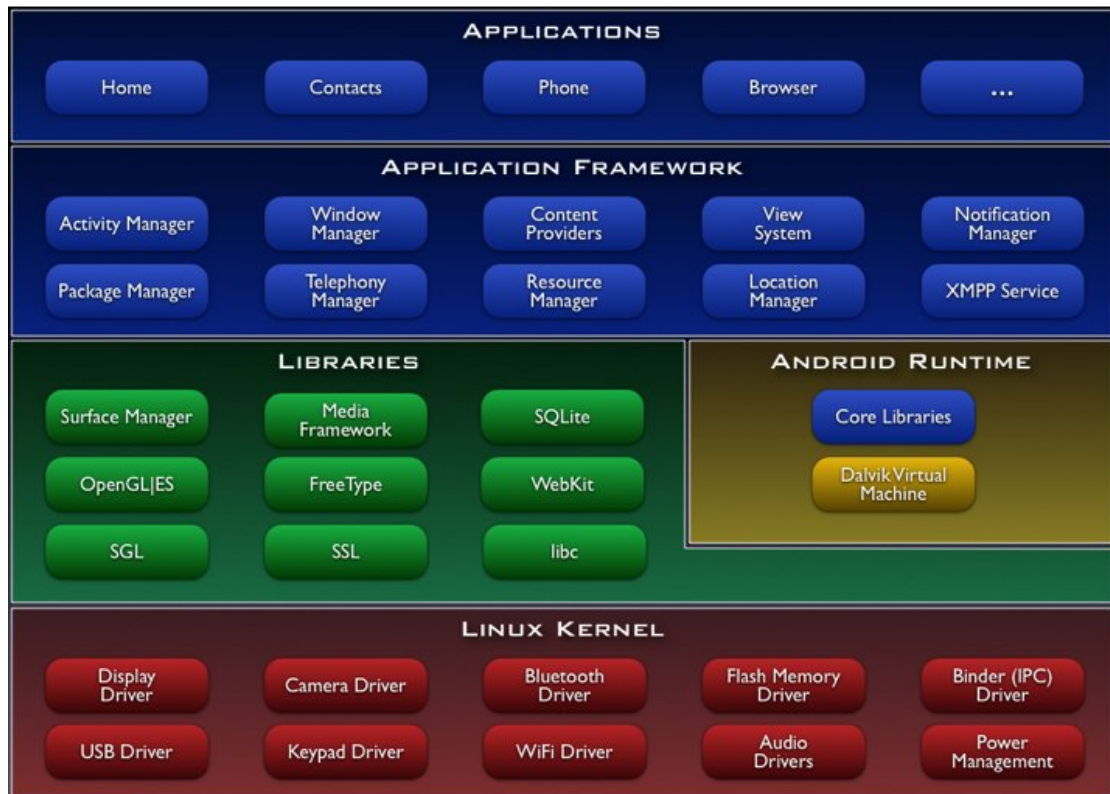


圖 2-1 Android Architecture⁵[12]

Android的主要特色[12]如下列所示：

1. Android 使用 Bionic Libc 作為系統的 C Library，它的特色是 BSD License[13]，大小約 200KB，更適合用在 Embedded Linux System。
2. 豐富的多媒體功能，支援 MPEG4、H.264、MP3、AAC、AMR、JPG and PNG。

⁵ 在撰寫本論文的過程中，原先參考網址 <http://code.google.com/android/what-is-android.html> 已被移除。

3. 支援 2D、3D 的顯示畫面。
4. 支援多國字型。
5. 輕量級的資料庫系統。
6. 現成的應用程式框架。
7. Dalvik Virtual Machine。
8. 豐富的網路功能，包含藍芽、無線網路等等。
9. 豐富的開發環境，包含Android Emulator、Debugging Tools、ADT (Android Development Tools) [14] for Eclipse[15]。圖 2-2為Android Emulator的執行畫面。

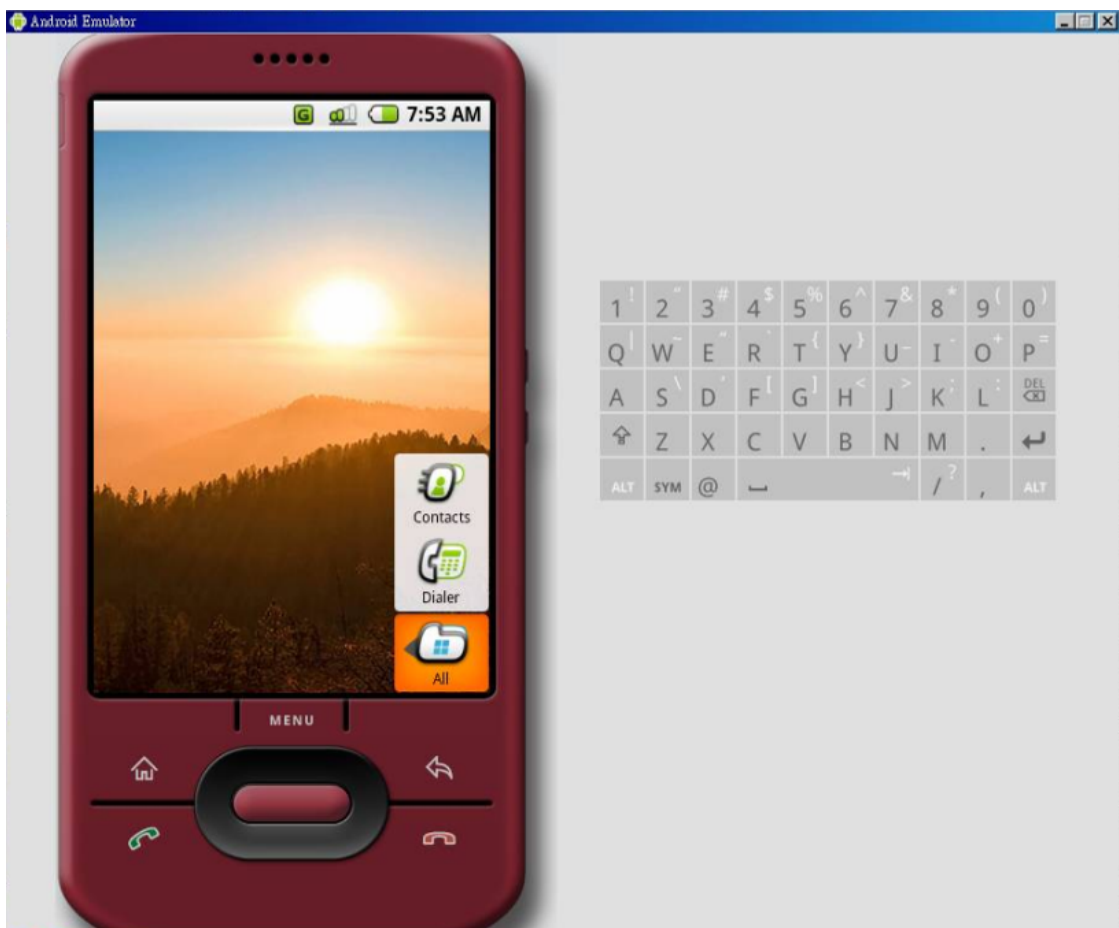


圖 2-2 Android Emulator

2.3 開發環境

Google 為 Android 提供完整的開發資源，包含 ADT、Android Emulator 等其它相關工具，開發人員可以使用 Eclipse 加上 ADT 來開發 Android 應用程式，而 Android Emulator 提供開發人員一個不需要實際硬體的測試環境，方便測試自行開發的 Android 相關程式，達到前期除錯、縮短開發時程等好處，Google 開發 Android Emulator 的另一個目的也是要加快全世界 Android developer 的開發速度。

Android SDK[16]中包含Android Emulator、Debugging Tools等其它相關開發工具，我們在此摘錄部份重點，Android SDK支援下列作業系統：

1. Windows⁶ XP (32-bit) or Vista (32- or 64-bit)
2. Mac OS X 10.4.8 or later (x86 only)
3. Linux (tested on Linux Ubuntu Dapper Drake)

Android SDK 支援下列開發環境：

1. Eclipse IDE。
2. Apache Ant 1.6.5 or later for Linux and Mac，1.7 or later for Windows。

自從Google在西元 2008 年 9 月正式公開Android 1.0 原始碼之後，我們可以從網路上直接下載原始碼來進行平台移植及相關開發工作，下載方式可以參考4.4或是<http://source.android.com/download/using-repo>⁷。

Android初期僅支援Qualcomm[17] MSM7K系列晶片，其它知名IC設計大廠如果要發展Android平台，必須自行投入研發人力進行移植的工作，且Android的目標是手持式裝置，也就是手機市場，初期尚無將觸角延伸至其它電子產品的計畫。

2.4 Linux手機聯盟

目前 Linux 手機相關的聯盟有三個：

⁶ 此處表示 Microsoft® Windows® 作業系統

⁷ Web 介面的原始碼瀏覽網址已經從 <http://git.android.com> 轉移至 <http://android.git.kernel.org/>

1. LiPS：起步最早，後來漸漸式微，便與後起 LiMo 合併。
2. LiMo[18]：主要成員為日系廠商、Motorola及Samsung。
3. OHA[19]：由Google所主導的新興聯盟，雖然各家廠商認為Google的力量很大，也有不少廠商加入，但是初期遲遲未見產品上市，原因是Google對於加入該聯盟的廠商所制定的規則非常嚴苛，廠商必須公開該產品的所有相關程式碼，且硬體規格一旦制定，未經Google許可，不能擅自改動任何元件，如此嚴苛的限制且不利於各家手機業者自行開發的方式阻礙了OHA的發展，也造成初期檯面上只有一家手機業者HTC宣布開發的時程及計畫，雖然Samsung及LG都有開發相關產品，但皆未有量產的計畫；但是在Android 1.0 正式公開之後，情況有所改變，世界各大手機代工廠及手機品牌大廠都紛紛投入研發人力，進行Android的相關開發工作。



第三章 一般Embedded Linux System簡介

3.1 系統架構

一般Embedded Linux System架構如圖 3-1。

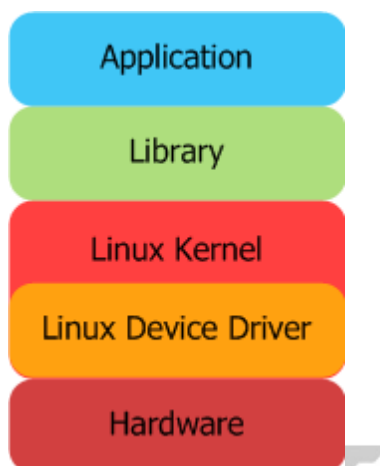


圖 3-1 一般 Embedded Linux System 架構

3.2 開機流程

一般Embedded Linux System開機流程如圖 3-2。Bootloader（一般Embedded Linux System常用的bootloader有許多種，包含blob、U-Boot等等）[20][21]是在執行Linux kernel前的一段程式，系統一開機由bootloader對CPU、記憶體、RS232、網路卡等相關硬體作初始化的設定，接著複製Flash ROM中的kernel image至記憶體，然後設置Linux kernel的啟動參數並開始執行記憶體中的Linux kernel，將CPU控制權交給Linux kernel。Linux kernel在執行完系統初始化設定之後，會將控制權轉移給user space的程式：init，在Embedded Linux的環境下，init一般為busybox⁸[22]，然後busybox會產生shell（附錄A）讓使用者可以透過特定指令與作業系統互動。

⁸ busybox 這個特殊的電腦用詞比較沒有強調大小寫的差異，所以在本論文中可見：Busybox、busybox、BusyBox 等不同大小寫表示法，但意義皆相同。

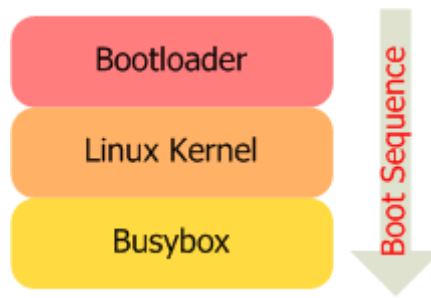


圖 3-2 一般 Embedded Linux System 開機流程

3.2.1 在ARM[23]架構下的開機流程

嵌入式系統一開機會由CPU讀取某固定位置的指令，而這個位置存放的即為bootloader的程式碼，當bootloader完成系統初始化而將控制權交給Linux kernel的同時，會將該平台的architecture number暫存在CPU register r1，由Linux kernel讀取CPU register r1 並與Linux kernel的architecture number進行比對，若是相同則繼續進行開機程序，若是不同則會在Uncompressing Linux kernel之後停住。Bootloader與Linux kernel的architecture number比對失敗將如圖 3-3所示。

```
Erasing sector 33 ... done
Erasing sector 34 ... done
Erasing sector 35 ... done
Erasing sector 36 ... done
Erasing sector 37 ... done
Erasing sector 38 ... done
Erased 28 sectors
u-boot$ cp.b a1100000 100000 200000
Copy to Flash...-done
u-boot$ boot
## Booting image at 00100000 ...
Image Name: EPS-Android
Created: 2009-06-07 5:01:03 UTC
Image Type: ARM Linux Kernel Image (gzip compressed)
Data Size: 1859912 Bytes = 1.8 MB
Load Address: a0008000
Entry Point: a0008000
Verifying Checksum ... OK
Uncompressing Kernel Image ... OK

Starting kernel ...

Uncompressing Linux.....
```

圖 3-3 Bootloader 與 Linux kernel 的 architecture number 比對失敗

若是希望得到更多的除錯資訊，可以透過 make menuconfig 指令開啟 Linux

kernel config 的 CONFIG_DEBUG_LL 選項：

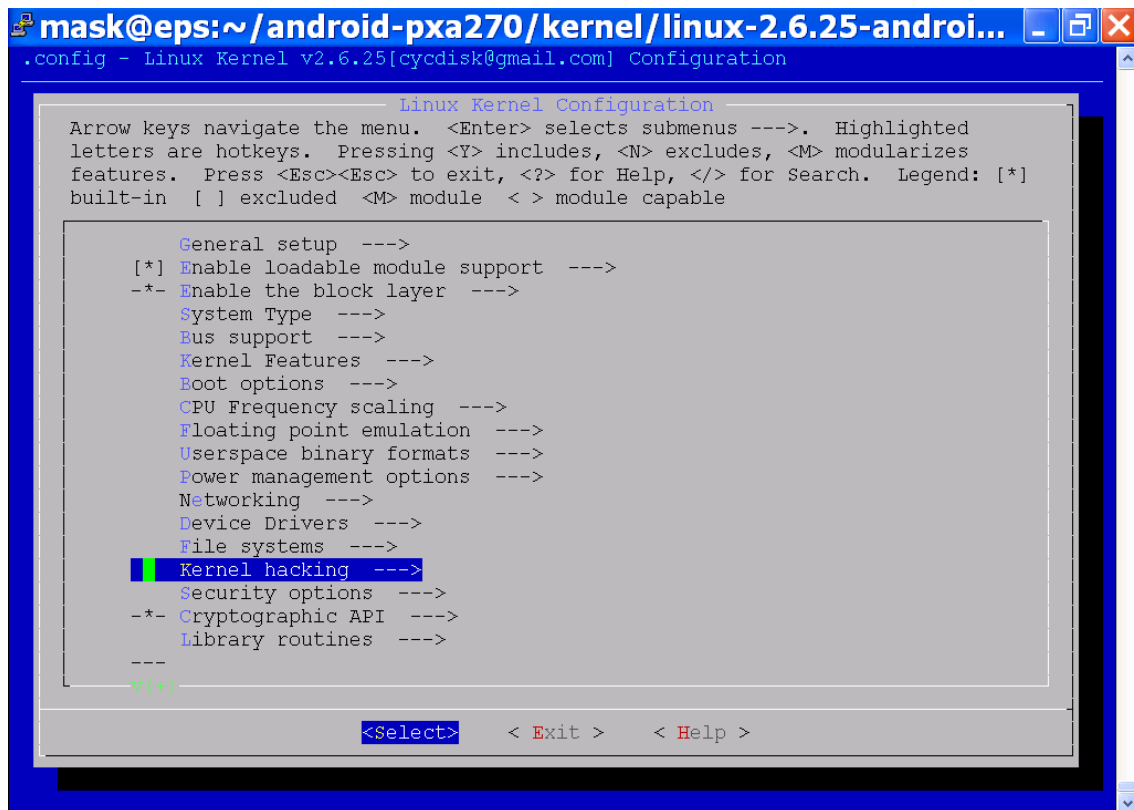


圖 3-4 選擇 Kernel hacking

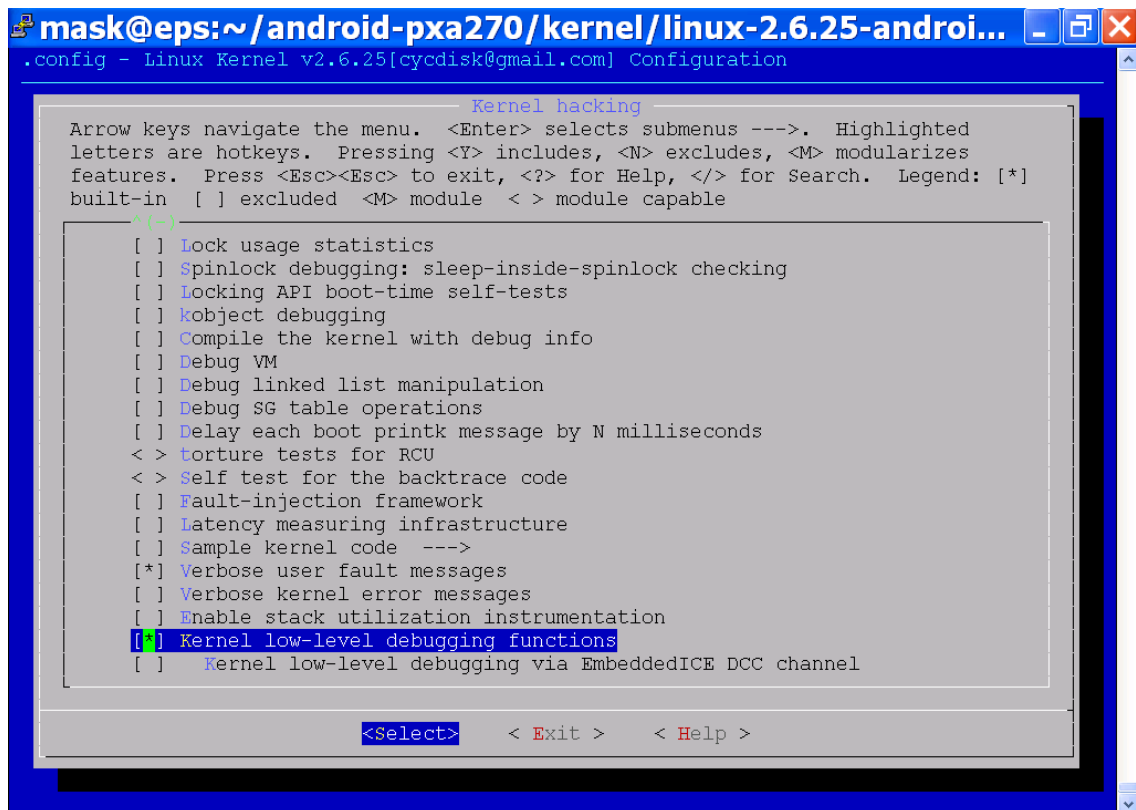


圖 3-5 開啟 CONFIG_DEBUG_LL 選項

圖 3-6為因為bootloader與Linux kernel的architecture number不同，在開啟Linux kernel config中的CONFIG_DEBUG_LL後所顯示的除錯訊息，0x3eb為bootloader記載的architecture number，而 0x2d9 為Linux kernel記載的architecture number。Bootloader存放architecture number的相關程式碼隨著bootloader不同而有所差異。Linux kernel會將architecture number存放在arch/arm/tools/mach-types，bootloader與Linux kernel的architecture number必須相同，否則無法順利開機，若是因為architecture number相異而無法開機，可以嘗試修改其中一方使其相同，或是參考我們4.2所使用的方法。

```

u-boot$ boot
## Booting image at 00100000 ...
Image Name:     EPS-Android
Created:        2009-06-07   4:57:30 UTC
Image Type:     ARM Linux Kernel Image (gzip compressed)
Data Size:      1860039 Bytes = 1.8 MB
Load Address:   a0008000
Entry Point:    a0008000
Verifying Checksum ... OK
Uncompressing Kernel Image ... OK

Starting kernel ...

Uncompressing Linux.....

Error: unrecognized/unsupported machine ID (r1 = 0x000003eb).

Available machine support:

ID (hex)      NAME
000002d9      XScale-PXA270 Module

Please check your kernel config and/or bootloader.

```

圖 3-6 因為 bootloader 與 Linux kernel 的 architecture number 不同，在開啟 Linux kernel config 中的 CONFIG_DEBUG_LL 後所顯示的除錯訊息

Linux kernel 開機流程如圖 3-7。一般跟（硬體）平台相關的設定會記錄在一個以平台名稱為檔名的.c檔中，我們暫時稱它為board.c。以我們的實驗平台為例，board.c的路徑是linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/mach-creator-pxa270.c，board.c的存放路徑說明：linux-kernel/與architecture相關/屬於arm平台/屬於machine-pxa系列/某平台的board.c。一般board.c主要是用來做硬體的相關設定，包含DMA、IRQ、GPIO、各Register等硬體相關設定。但是畢竟一個檔案可能無法涵蓋該平台的所有硬體設定，所以通常有部分與硬體設定相關的程式碼會放在其它相關的檔案中，這部分視平台而有所不同。

Linux kernel 在執行完系統初始化設定（包含設置記憶體、Scheduler、中斷向量表、載入驅動程式等等）之後，會將控制權以 kernel thread 的形式轉移給 user space 的第一個程式：init，在 Embedded Linux 的環境下，init 一般包含於 busybox 中。init執行流程如圖 3-8。當Linux kernel將控制權轉移給init時，init會讀取/etc⁹下的

⁹ 對 Busybox 而言，/etc 是檔案系統上最重要的目錄，因為 Busybox 會讀取/etc 下的相關檔案來作系統環境的設定。

相關檔案，一般是讀取/etc/inittab，再由/etc/inittab引入/etc/init.d/rcS等相關檔案執行系統環境的設定及初始化的動作，包含掛載檔案系統、執行相關應用程式及產生shell讓使用者可以透過特定指令與作業系統互動；busybox透過以上的流程來完成系統最後的相關設定。

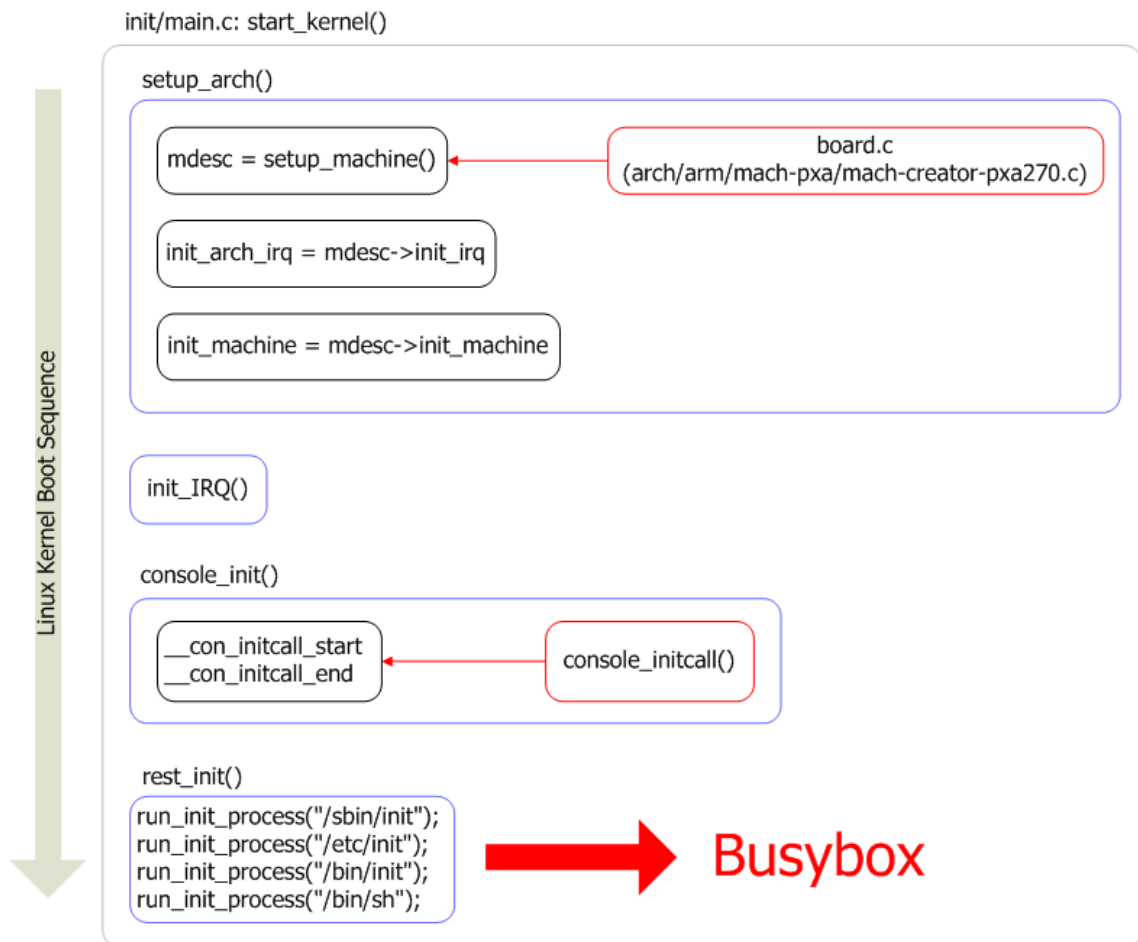


圖 3-7 Linux kernel 開機流程

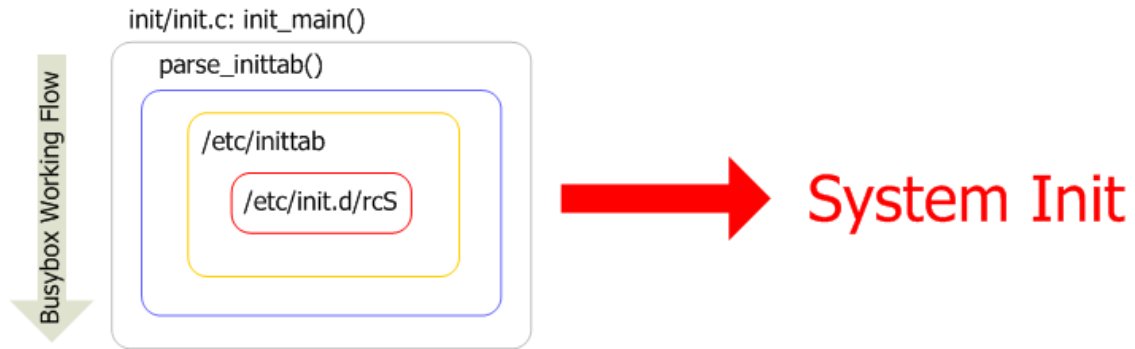


圖 3-8 init 執行流程

一般Embedded Linux System的目錄結構如圖 3-9，以下為各目錄的說明：

/bin，/sbin：一些常用指令及工具程式。

/dev：device node(在Linux下，device node為一種特殊檔案，一般Linux device driver會建立device node，而user space透過device node來操作device driver進而達到控制硬體的目的)。

/etc：包含系統設定檔及開機所需的Init Script(在此Script一般是利用Shell語法寫成的程式碼，由Shell來翻譯並執行，可以參考附錄A)。

/lib：包含Linux kernel Modules (Linux device driver) 及動態連結檔(.so¹⁰)。

/proc：此目錄的用意是提供系統上所有Process的相關資訊；而Linux device driver不應該透過此目錄存取device driver的相關資訊，應該透過/sys存取device driver相關資訊。

/root：額外所需的工具程式及檔案。

/sys：1. Linux kernel提供一個讓使用者讀取/寫入Linux kernel資訊的操作介面。一般Linux device driver會在此提供存取介面讓使用者可以透過此介面存取Linux device driver的相關資訊。

2. 記錄整個系統及周邊I/O的硬體資訊。

¹⁰ 在Linux下，一般動態連結檔的附檔名為so，動態連結檔可以被執行檔或是其它動態連結檔動態地載入並執行，部份以Embedded Linux System為基礎的電子產品是透過更新動態連結檔的方式來更新系統。

/var：存放一般的 log。

/usr：視需要而存在。

/tmp：一般為記憶體空間，所存放的檔案在關機後就會消失，所以若是重要的檔案請勿放在此目錄下。

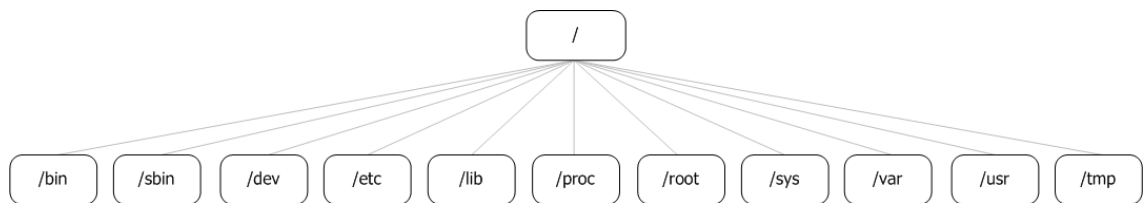


圖 3-9 一般 Embedded Linux System 的目錄結構

只要熟知本節內容就具備了建構一 Pure Embedded Linux System 的條件。一般 Embedded Linux System 的建構流程如圖 3-10。

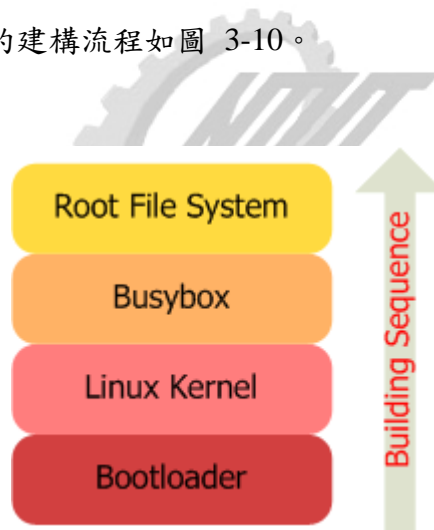


圖 3-10 一般 Embedded Linux System 的建構流程

3.3 開發流程

圖 3-11 為我們的開發環境。我們使用 target / host 的開發方式：

1. Target 端：指嵌入式系統
2. Host 端：一般為個人電腦或是工作站

目前一般嵌入式開發環境為 target / host 的架構，為什麼會有這種架構出現？

因為一般嵌入式系統的資源有限，包含 CPU 速度較慢，記憶體較小，缺乏編譯器及系統軟體等等，且部分軟體執行時需要虛擬記憶體，嵌入式系統也許沒有虛擬記憶體的機制，所以目前一般都是 target / host 的架構，使用足夠資源的機器(host)作開發，再將開發好的映像檔寫入嵌入式系統(target)中，如此可加快開發速度、縮短開發時程。



圖 3-11 我們的開發環境

第四章 Android移植步驟及流程¹¹

我們移植的目標為建構一個可以實際執行 Android 的環境，包含看見 Android 的畫面、進行 Android 的基本操作，例如 Keypad、Touch 等等，至於進一步的系統微調及因為不同平台所需進行的細部修改，本論文不多作說明。

本章撰述的方式會先說明大方向，緊接著詳述細節的部分，包含修改的檔案及程式碼的介紹等等。

4.1 實驗平台 PXA270 的硬體概況

Android 的執行環境有其硬體限制，並不是任何硬體都可以成功地執行 Android，這部份請參考 2.1。



¹¹ 本章撰述的內容皆為我們實際開發此平台所採用的方式

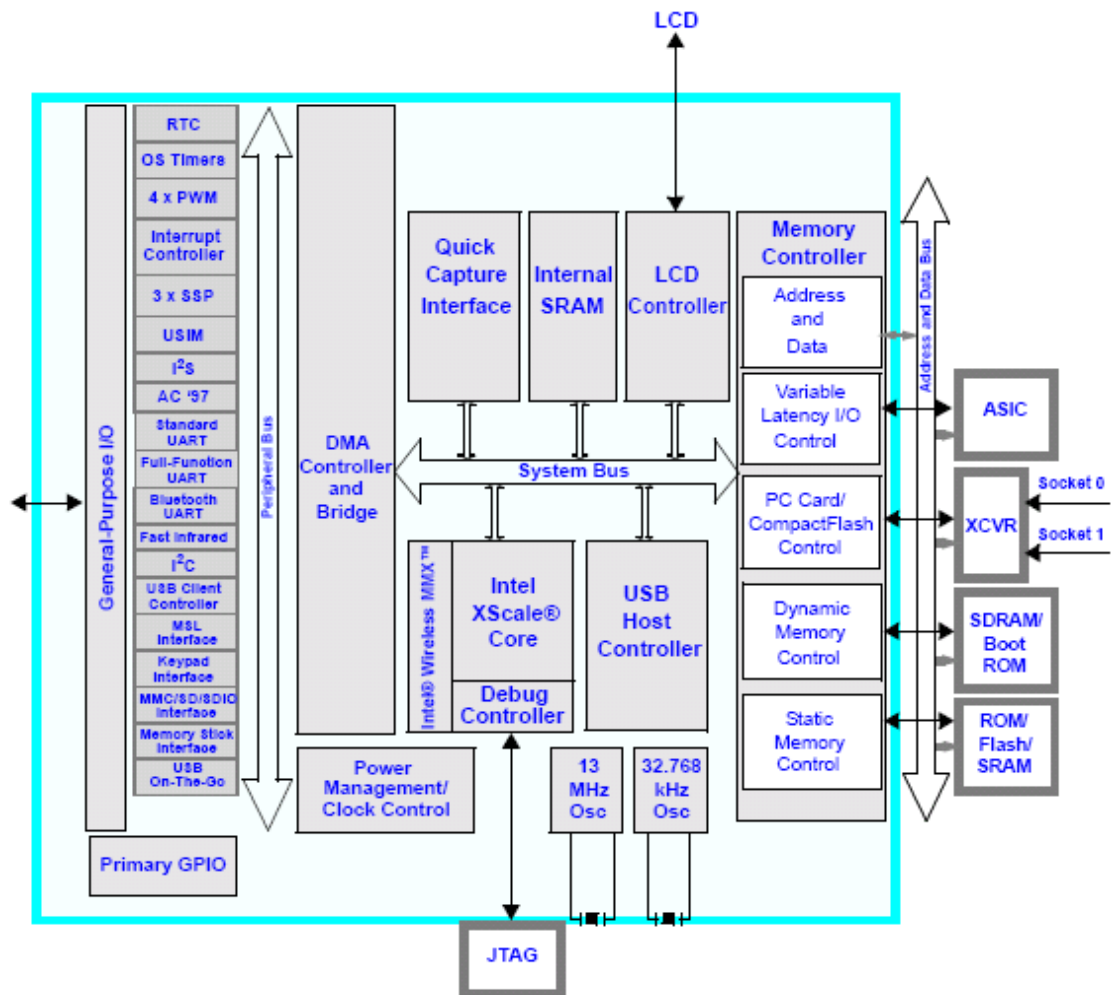


圖 4-1 PXA27x Processor Block Diagram¹²

¹² 摘錄自 Intel® PXA27x Processor Family Optimization Guide, August 2004. Order Number: 280004-002.

表 4-1 實驗平台 PXA270 硬體規格

CPU	Intel XScale PXA270 520MHz	
Flash ROM	32M Bytes	
SDRAM	64M Bytes	
keypad	4x4 matrix	
touch	UCB1400	
LCD Module(LCM)	LCD Panel	TOPPOLY TD035STEB1
	Display Area	53.64mm(H) x 71.52mm(V)
	Drive System	TFT active matrix
	Display Colors	262144 colors
	Number of Pixels	240 x RGB(H) x 320(V)
	Pixel Arrangement	RGB Vertical stripe
	Signal System	6-bit digital signals for each RGB
UART		
Ethernet	10/100 Mbps	
USB		
Audio	UCB1400	

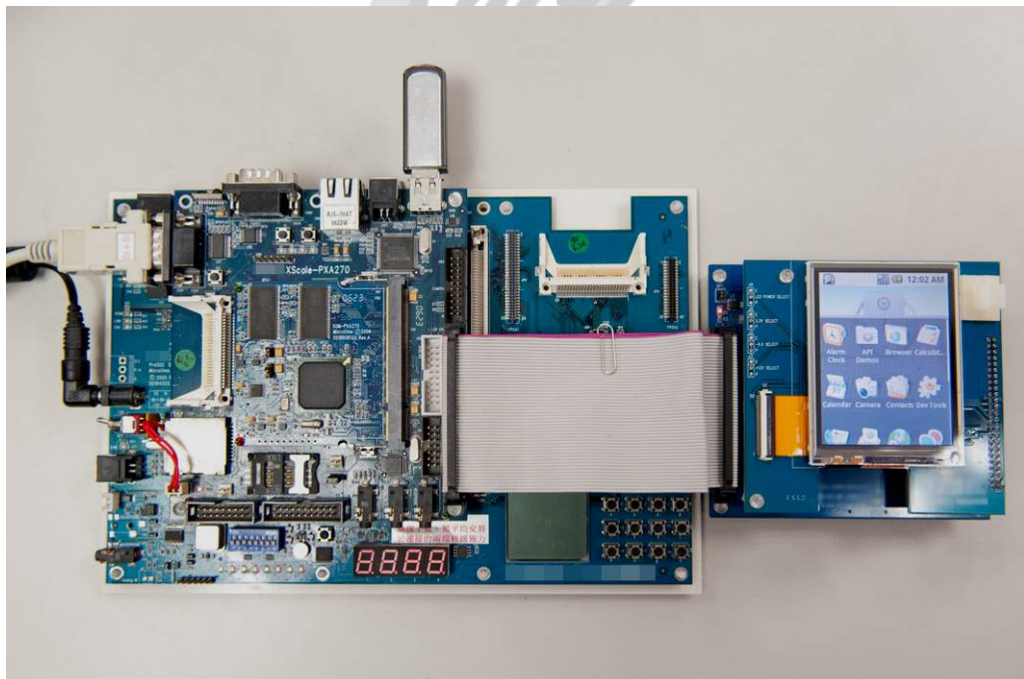


圖 4-2 實驗平台 PXA270

4.2 準備工作

在Google尚未公開Android原始碼之前，我們可以透過Android Emulator取得Android執行環境的二進位檔，並修改Linux device driver來配合Android library及Android application的行為，但此方式並非正統的移植方法，這種類似hacking再加上try and error的方法礙於不確定性及side effects，因此我們不對此方法多作說明¹³，本章是在能夠直接下載Android原始碼的條件下所撰述。

我們的目標是先建構一個Pure Embedded Linux System，接著建立Android Runtime System，我們這樣做的原因是希望此平台不僅能夠執行Android，亦可以是一個單純的開發環境，能夠執行一般透過toolchain所cross compile的程式，就像以往開發嵌入式系統一樣，讓使用者從簡單的Embedded Linux System逐步進入Android的世界，最後能夠在我們的實驗平台上開發Android相關程式。所以在正式介紹如何移植Android之前，我們需要對Embedded Linux System有一個綜觀的了解，需要知道Embedded Linux System的building block、如何建構一個Embedded Linux System、Embedded Linux System的開機流程、控制權的轉移、檔案系統的架構等等；如此我們才能夠從底層往上建構這樣的開發環境，這部分可以參考3.2。

在此我們不對bootloader作太多說明，因為移植過程中我們沒有對硬體製造商提供的bootloader作任何改動；因為bootloader與硬體有相當高的關聯性，所以一般嵌入式系統都會對bootloader作一定程度的修改，但若是單就移植Android來說，是不需要修改到bootloader的部分。簡單地說bootloader只是將CPU控制權交到OS手中，唯一要注意的是Linux kernel會比對由bootloader傳入的architecture number，這部分的細節可以參考3.2，而我們使用的方法是直接修改linux-2.6.25-android-1.0_r1/arch/arm/kernel/head.S（附錄J）中CPU register r1 的值以符合 linux-2.6.25-android-1.0_r1/arch/arm/tools/mach-types 的設定，而忽略由bootloader傳入的Architecture Number。

¹³ 本論文是累積過去的移植經驗而衍生的移植流程，我們在PXA270的移植過程中，大多直接追蹤原始碼來解決所遇到的問題；但是我們在附錄W列出過去參考過的相關資料。

一般開發環境必須先行安裝部份套件才能夠順利地編譯Android，這部份視開發平台（如Fedora、CentOS、Debian、Ubuntu等不同的Linux distribution）而可能有不同的套件名稱，我們不在此多作贅述[24]。

下列為移植的關鍵：

1. 細心：謹慎的態度決定移植的成功與否。
2. 運氣：包含所使用的硬體是否夠受歡迎、能見度夠高、新版 Linux kernel 是否支援其 device driver。
3. 經驗：是否有過類似的移植經驗、對 Linux kernel、Linux device model、Linux device driver、Linux 操作環境、嵌入式系統的建置及 Linux 開機流程的熟稔程度。

一般 Embedded Linux System 的軟體元件大致分為：

1. Linux kernel
2. Library
3. Application

加上我們參考Android的軟體架構圖，所以我們將Android移植的步驟分為下列兩步，如圖 4-3所示：

1. 移植 Linux kernel
2. 整合性修改

其中 1，我們採用 2008 年 9 月的Linux 2.6.25 for Android 1.0 SDK, Release 1：linux-2.6.25-android-1.0_r1¹⁴，因為linux-2.6.25-android-1.0_r1 為我們整個移植的根基，所以務必謹慎小心，盡可能將硬體相關的程式碼完全移植至 linux-2.6.25-android-1.0_r1。

而 2 的部分，包含在成功移植 Linux kernel 的條件下，對 Linux device driver、Android 中的 Library、Application Framework 及 Application 等部份作一定程度的修

¹⁴ 可於 <http://code.google.com/p/android/downloads/list> 或 http://android.googlecode.com/files/linux-2.6.25-android-1.0_r1.tar.gz 下載

改，達到可以正常執行 Android 的目的，這部分視平台不同或需要作相關修改。

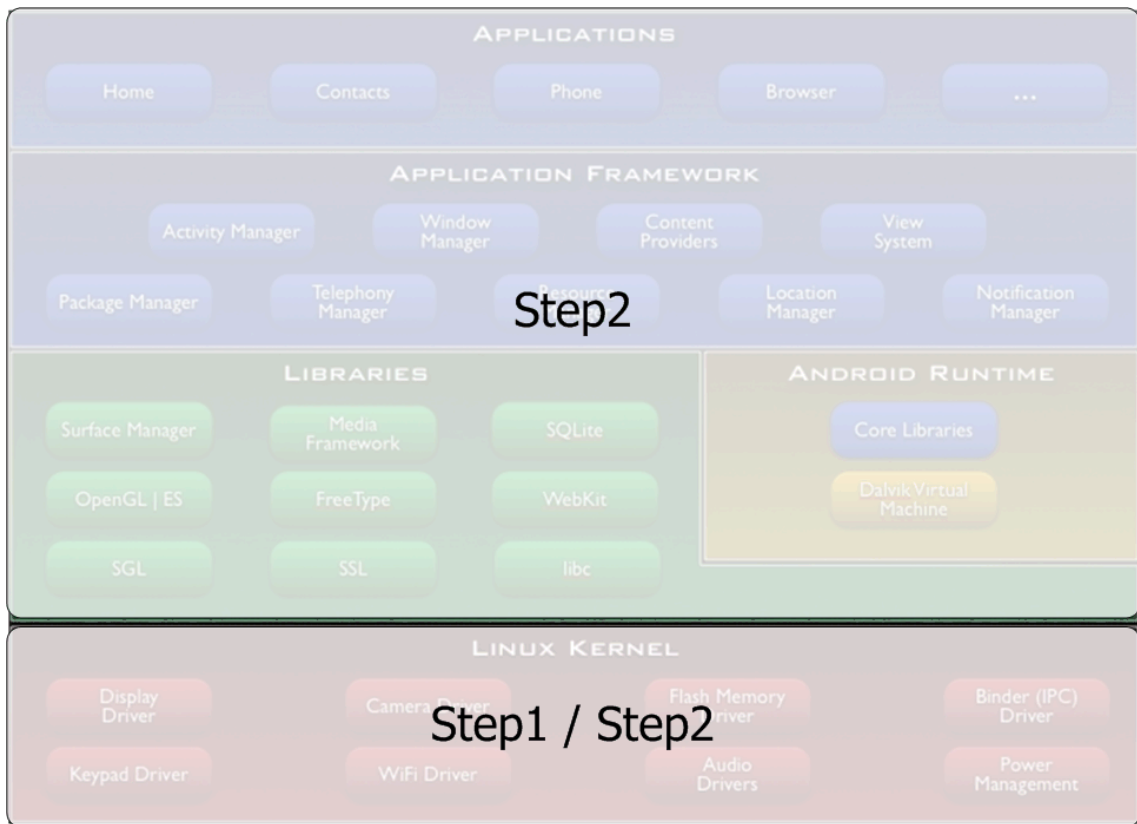


圖 4-3 Android 移植步驟

4.3 移植Linux kernel

我們購置PXA270 的時間為 2006~2007 年間，當時硬體廠商提供的Linux kernel為linux-2.6.15.3，因此不可能包含任何跟Android相關的程式碼。基於Android對於Linux kernel的限制，我們採用linux-2.6.25-android-1.0_r1，因為linux-2.6.15.3與linux-2.6.25-android-1.0_r1 的版本差異太大，所以我們使用人工patch的方式，4.3.1將介紹人工patch的工具。我們移植的原則是拿硬體廠商所提供的linux-2.6.15.3與 <http://www.kernel.org> 的 linux-2.6.15.3 作比較，將差異的部份加入linux-2.6.25-android-1.0_r1 中，我們所使用的方法可能不能稱為SOP（Standard Operating Procedure），因為移植的過程在不同的平台、不同的環境下有不同的方

法，若是硬體廠商所提供的Linux kernel與Android所支援的版本相近，也許可以直接使用patch（一個Linux的指令，將新版與舊版程式碼的差異套用於舊版程式碼以更新舊版程式碼）的方式來移植Linux kernel。在我們的例子中，我們採用的方法是在現有條件下最節省時間、最不浪費人力及資源的方式，原因是在移植Linux kernel的部分，我們所採用的方法只需要集中精神在硬體相關(Hardware dependent)的部分，我們幾乎沒有浪費時間在Android對Linux kernel所加入的patch（單就移植過程而言，不包含日後開發的部分）。圖 4-4為我們移植Linux kernel的流程圖。

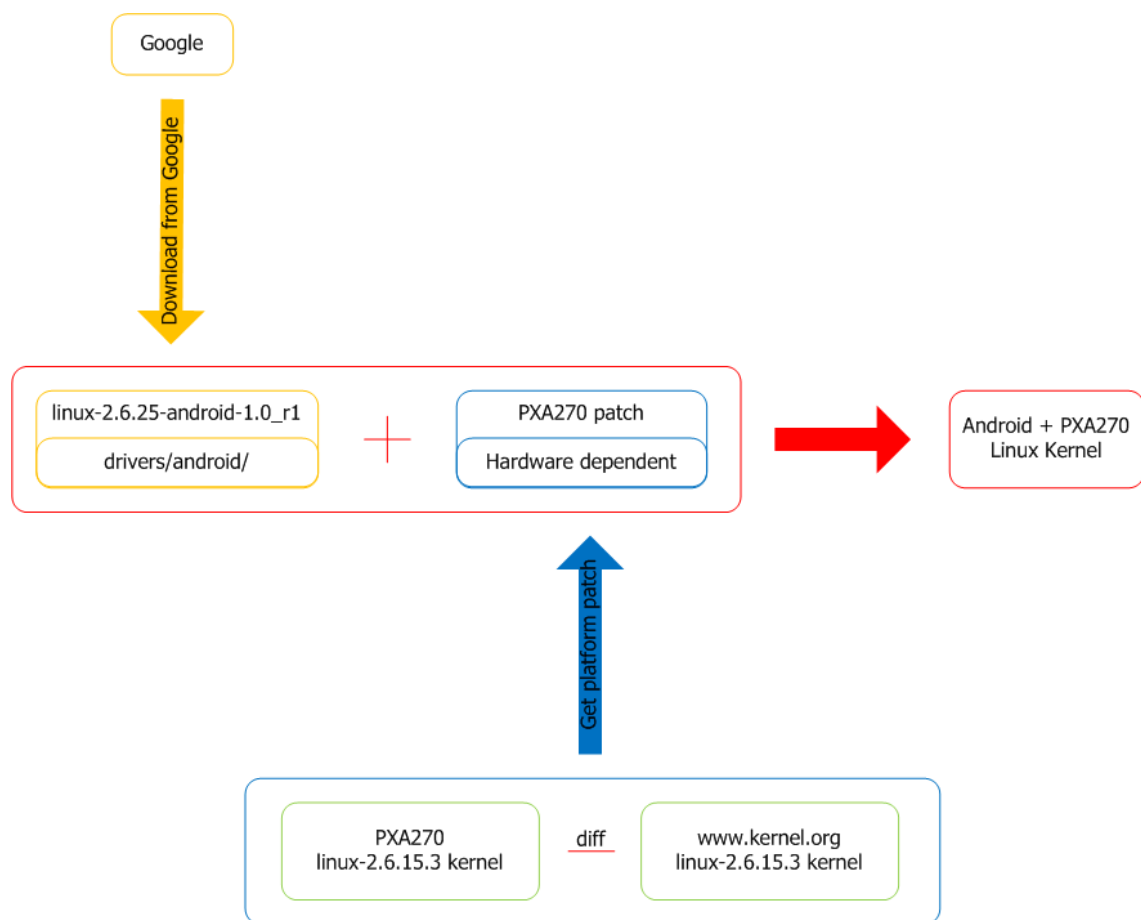


圖 4-4 移植 Linux kernel 的流程圖

移植 Linux kernel 可以分為下列兩步：

1. 目標為可以開機及看到 console 訊息。
2. 以整合性修改為主，為搭配 Android library 及 Android application 等所修

改周邊裝置的驅動程式。

本節我們僅討論 1 的部分，而 2 的部分請參考4.4。

4.3.1 使用工具

我們使用圖形化界面的比較工具進行人工移植的程序，因為圖形化界面比較友善且容易使用，在移植過程中可能需要對大量檔案進行比較的動作，若是使用傳統的 diff 等工具可能不是那麼地直覺且不易閱讀，這些圖形化比較工具在內部依然是使用傳統的 diff 或是類似 diff 的工具，只是在前端加進圖形化界面以方便使用者閱讀。在此我們介紹下列兩種圖形化界面的比較工具：

1. Windows：WinMerge[25]
2. Linux：Meld[26]

若是以Windows為開發環境，可以使用WinMerge；若是以Linux為開發環境，可以使用Meld。這兩套軟體為圖形化介面的比較及合併工具，且皆為open source software，我們使用的是WinMerge。圖 4-5為WinMerge。圖 4-6為Meld。

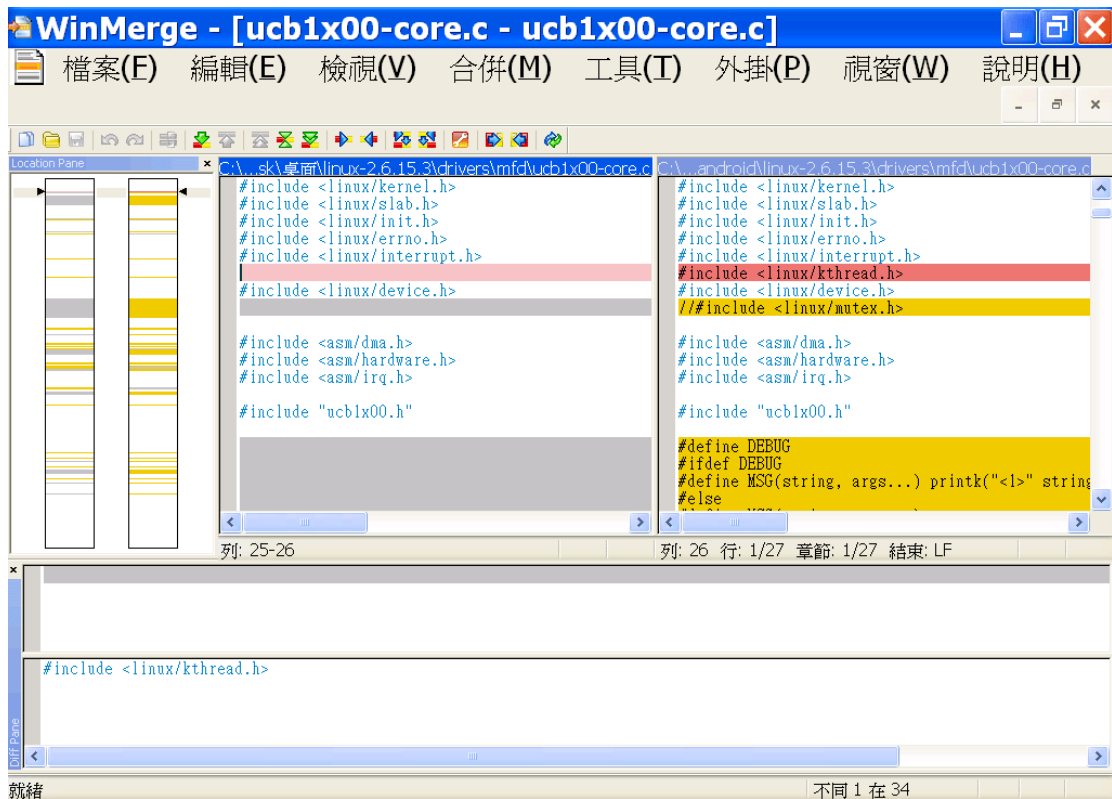


圖 4-5 WinMerge

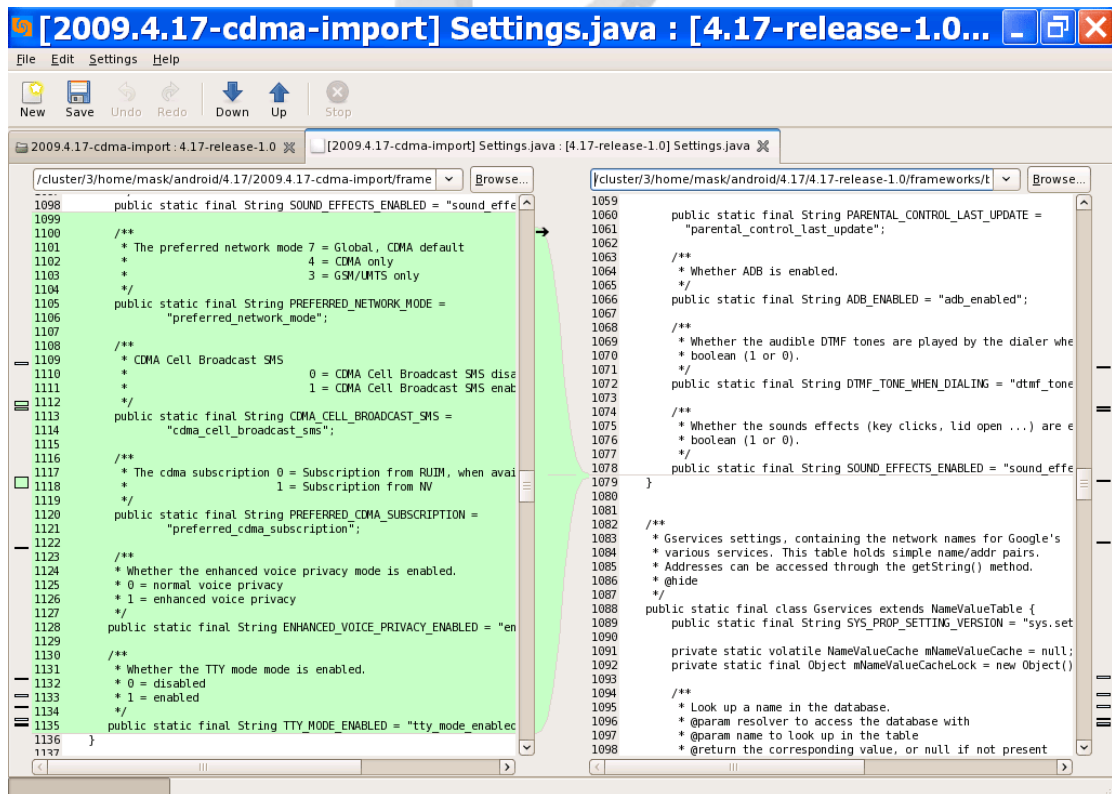


圖 4-6 Meld

4.3.2 可能遇到的問題及解決方法

我們以 PXA270 為例，加上過去移植其它平台的經驗，來說明移植的原則，針對 Linux kernel 的部份，因為新舊版本（在此新版為 linux-2.6.25-android-1.0_r1，舊版為 linux-2.6.15.3）之間的差異而可能遇到的問題，我們提出可能的解決方法。當真正看到兩份程式碼的差異時，剎那間可能會不知所措，因為無法預期可能會發生的狀況，這時要靜下心來，把握下列原則盡可能地移植：

1. 盡可能將（硬體）平台相關（包含 CPU 及周邊硬體等設定）的程式碼從舊版移植至新版。
2. 針對硬體相關的程式碼，如果舊版改動之處（拿我們的 linux-2.6.15.3 與 <http://www.kernel.org> 的 linux-2.6.15.3 作比較）與新版相同，但是修改的內容不同，則以舊版的為主，因為舊版是真正可以運作的版本，代表與硬體相關的程式碼是沒有問題的。
3. 非硬體相關的程式碼，盡可能使用新版的。
4. 若是遇到不知如何處理的情況，先跳過，記錄該檔名，先完成比較好移植及有把握的檔案，最後再回過頭來處理先前跳過的檔案，這部份可能需要參考部份相關原始碼才能決定要如何移植：
 - 參考其它同樣使用 PXA270 的平台，比較新舊版本的差異，我們是參考 Mainstone 等平台的相關程式碼。
 - 參考其它同樣使用 PXA270 且可以在 linux-2.6.25 下運作的平台，觀察它們所修改的程式碼。
 - 部份Linux kernel或是device driver的原始碼有大幅度變動時，包含以新檔名取代舊檔、資料結構改變等，這部份可能要比較多個檔案才能決定要如何修改，我們建議使用ctags[27]，靜態追蹤原始碼所修改的部份，並謹慎的判斷真正需要移植的程式碼。

上述四點為我們移植的原則。在我們移植的過程中，遇到的問題及解決方法

大致可分為下列五項：

1. 資料結構改變：參考其它相同或是類似的平台，比較新舊版本的差異，我們是參考 Mainstone 等平台的相關程式碼。
2. Init Section 改變：一般發生在 device driver，參考其它在新版編譯成功的 device driver，觀察它們從舊版到新版所作的修改，仿照它們，修改編譯發生問題的 device driver。
3. 程式碼被散佈在多個檔案中：這項一般伴隨著下一項。就我們的觀察，Linux kernel 的眾多維護者對檔案命名及檔案的分佈位置有很嚴謹的考量，某些可能說不上是規範或是規定，但卻都是直觀且可以理解的。所以程式碼的位置在不同的版本可能有很大的變化。解決方法是只考慮舊版所改動的部分，就是拿我們的 linux-2.6.15.3 與 <http://www.kernel.org> 的 linux-2.6.15.3 作比較，將修改的部分套用至新版中，在此要注意 redefine、undefine、function renamed、function prototype modified 等問題，這部份要非常細心，可能要參考許多檔案才能知道有哪些地方是需要修改的。
4. 檔案名稱改變：這項一般伴隨著上一項。解決方法同上。
5. 如何得到新的 kernel config (.config，Linux kernel 的設定檔)，有兩個方法可以得到新的 kernel konfig：
 - make oldconfig：使用舊的.config 產生新的.config。
 - make pxa270_old_defconfig：直接指定 defconfig 檔，這邊假設 pxa270_old_defconfig 為我們舊版的 defconfig，將該檔案複製至 linux-2.6.25-android-1.0_r1/arch/arm/configs/下。

我們無法預期因為平台不同而發生的狀況，因此上述方法無法保證能夠解決各種平台所發生的問題。

4.3.3 編譯 Linux kernel

因為自行製作 toolchain 的過程頗為複雜，既然我們正在使用及開發 open source software，我們要利用這個優點，站在巨人的肩膀上，直接使用免費且功能符合我們需求的 toolchain：<http://www.codesourcery.com/sgpp/lite/arm/portal/package3397/public/arm-none-linux-gnueabi/arm-2008q3-41-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2>。

通常移植後的第一次編譯都會發生錯誤，此時我們使用的方法是針對錯誤訊息進行相關的修改（參考4.3.2），如果不知道如何修改或是該錯誤不影響Linux kernel的開機流程，可以選擇直接disable相關功能，目的就是要想盡辦法讓移植的Linux kernel可以開機並看見console的訊息。

4.3.4 測試 Linux kernel

在開發初期，kernel 尚未穩定的情況下，為了縮短開發時間，盡量直接透過網路將 kernel image 載入記憶體中並執行，不需要寫入 Flash ROM 中。

若是遇到 kernel panic，試著從錯誤訊息中找出無法順利開機的原因，搭配 printk 找出真正發生問題的程式碼，依個人需求，嘗試修改或是直接 disable 該功能。

在開發初期，建議使用NFS[28]掛載root file system，若是遇到root file system無法成功掛載，可以參考3.2.1，先建構一般root file system的目錄結構，然後使用編譯kernel的toolchain編譯Busybox，將編譯成功的busybox加入root file system中，並在etc下加入所需的檔案，如此應該可以正常開機。

4.4 整合性修改

在成功移植 Linux kernel 之後，接著要對 Linux kernel、Linux device driver、Android library、Android application 及 Android 執行環境等作整合性的修改。程式

碼的修改視實驗平台、硬體環境及所遇到的問題而不同。我們修改的原則是以最少的修改，並盡可能以可讀性且最少的 side effects 來達到預期的結果。

我們在此的修改方式僅供參考，因為不同的人移植不同平台所遇到的問題可能不盡相同，而與我們遇到相同問題的人可能不多，因為一是此實驗平台購置已久，二是一般硬體廠商會協助解決相關問題等等。

在此我們必須先下載 Android 原始碼，目前 Android 原始碼的大小約 2~3GB，下載原始碼所需的時間視個人網路狀況而定，我們使用 cdma-import 的版本，在此必須確認完整下載 Android 原始碼，以下是我們的操作步驟（詳細流程可以參考 <http://source.android.com/download>）：

1. `curl http://android.git.kernel.org/repo > repo`
2. `chmod +x repo`
3. `./repo init -u git://android.git.kernel.org/platform/manifest.git -b cdma-import`
4. `./repo sync`

我們在開發過程中遇到三個比較嚴重的問題：

1. LCD：畫面閃爍。
2. Keypad：缺少 driver。
3. Touch：更換 driver。確認 device driver 運作正常之後，不論我們觸碰 touch 的任何位置，其總是反應出我們觸碰到 (0, 0)。

我們將上述問題歸納為下列兩個部份並獨立解決：

1. Device driver 的部分。
2. Android library、Android application 及 Android 執行環境等等。

1 的部份，因為我們可能需要修改部分 Linux device driver 的程式碼，例如沒有設定 IRQ、device driver 註冊失敗等等，所以必須自行加入部份程式碼使之可以正常運作，這部份需要相關經驗及背景知識，如果不知道從何著手，可以參考同

類型且正常運作的 device driver。

2 的部份視所遇到的問題而有不同的解決方法。接著我們將針對上述所遇到的三個問題，詳述其解決方法。

4.4.1 LCD

一般在成功移植 Linux kernel 且可以正常開機的情況下，執行 Android 會看不到 LCD 的顯示畫面，原因是 Android 需要 Double Framebuffer，所以我們必須先修改 Framebuffer driver 以獲得兩倍 Framebuffer 的記憶體，接著針對不同平台解決可能發生的問題。我們在 LCD 的部份遇到兩個問題：

1. Double Framebuffer
2. LCD 畫面閃爍

4.4.1.1 Double Framebuffer

一般在 Embedded Linux System 下是透過 Framebuffer 的機制來控制 LCD 的顯示畫面，簡單地說 LCD 的顯示畫面即為 Framebuffer 的內容，Framebuffer 對記憶體的使用量如 (4.1)。

$$Xresolution \times Yresolution \times \frac{Bits}{Pixel} \div \frac{Bits}{Byte} = FramebufferSize \quad (4.1)$$

Android 使用 Double Framebuffer，也就是在記憶體中存放兩倍 LCD 大小的畫面，我們猜測如此的設計不外乎是為了畫面的滑動、pre-draw 的機制、Swap Framebuffer 等效能上的考量，再加上 Framebuffer 對記憶體的使用量不大，所以 Double Framebuffer 並不會對系統資源造成太大影響。以我們的實驗平台為例，原本 Framebuffer 使用 $320 \times 240 \times 16 \div 8B = 153600B \div 0.15MB$ 的記憶體，使用 Double Framebuffer 後約使用 0.3MB 的記憶體，但我們的實驗平台是 2006~2007 年的產品，以目前的智慧型手機而言，0.3MB 的需求不會對系統資源造成太大影響。

針對 LCD 在 Android 的環境下無法顯示畫面的問題，Framebuffer Driver 必須向

Linux kernel要求兩倍Framebuffer的記憶體，如果一開始不知道從何修改且毫無頭緒，可以參考 Android Emulator 所使用的 Framebuffer Driver：
linux-2.6.25-android-1.0_r1/drivers/video/goldfishfb.c。我們實驗平台的Framebuffer driver為linux-2.6.25-android-1.0_r1/drivers/video/pxafb.c（附錄B）。

4.4.1.2 LCD 畫面閃爍

LCD畫面閃爍的問題如圖 4-7。LCD畫面閃爍是因為LCD controller不斷重複地開關所造成，而其不斷重複地開關是因為Framebuffer driver中的LCCR0(暫存器)設定值沒有同步，所以LCD的狀態不斷地切換所造成。

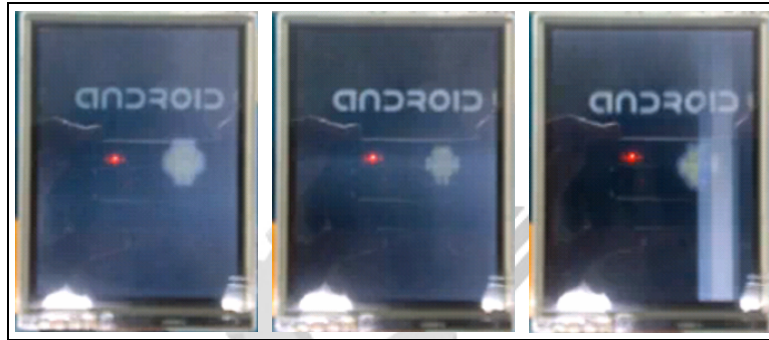


圖 4-7 LCD 畫面閃爍，由左至右每張圖像拍攝的時間間隔為 1 秒

我們的解決方法是增加fb_pan_display函式以更改LCD的狀態。我們實驗平台的Framebuffer driver程式碼路徑為linux-2.6.25-android-1.0_r1/drivers/video/pxafb.c（附錄B）。

4.4.2 Keypad

我們在 keypad 所遇到的問題及解決方法如下：

1. 我們實驗平台缺少 keypad driver，解決方法為新增我們實驗平台的 keypad driver（參考 linux-2.6.25-android-1.0_r1/drivers/input/keyboard/pxa27x_keypad.c，並作大幅度的修改）。
2. 我們實驗平台僅有 4x4 matrix key，解決方法為使用組合鍵來擴增按鍵

數，但儘管使用組合建的方式，按鍵數還是不夠，所以我們只選取某些特定或重要的按鍵值，而忽略部分按鍵值。

3. 我們實驗平台的 keypad 沒有硬體中斷，解決方法為使用 polling 的方式，但是我們考量低功耗及電源管理在行動裝置上是一項值得探討的議題，因此我們實作一個有效的演算法來偵測 keypad 被按壓的情形，並適當地降低 polling 頻率以達到降低功耗及省電的效果。如何在 keypad 被按壓時能夠即時反應，反之降低 polling 的頻率以減少耗電量？我們使用一個簡單的演算法，當 keypad 被按壓時，我們將 polling 的時間間隔設定成我們預設的最小間隔，在可能被按壓的時間區塊中集中偵測，但是如果在間隔時間內未被按壓，我們將逐步增加 polling 的時間間隔直到我們預設的最大時間間隔，如此在被按壓可能性較小的時間區塊中達到省電的效果。

我們實驗平台的 keypad driver 程式碼路徑為 linux-2.6.25-android-1.0_r1/drivers/input/keyboard/android_keypad.c

4.4.3 Touch

因為我們所使用的 touch driver 在新版與舊版 Linux kernel 之間有很大的差異，所以首先要確認新版 Linux kernel 中的 touch driver 可以正常運作，接著觀察 touch driver 在 Android 的環境下是否有其它問題。我們使用 HTC Dream 作為相關問題的對照組。

我們修正 touch driver 的四個問題：

1. 指定 IRQ
2. Connect to input subsystem
3. 校正從硬體讀取的 (X, Y) 座標值
4. 調整觸碰的靈敏度

我們使用 printk 驗證 touch driver 所得到的座標值，在此我們需要知道一個

touch event 會包含：

1. X 座標值
2. Y 座標值
3. EV_SYN¹⁵（表示完成一個包含X、Y座標值的touch event）

在touch driver運作正常的情況下，Android application對touch的反應如圖4-8，不論我們觸碰touch的任何位置，其得到的座標值似乎總是（0，0），於是我們追蹤touch event在Android的運作流程。

我們解決此問題的分析流程如下：

1. touch driver 是否正確將（X，Y）回報至 Android application？
2. Android application 是否收到正確的（X，Y）？
3. 若是2成立，是否因為某些條件造成（X，Y）被改變？

圖 4-10為在Android環境下，輸入裝置與電源管理的關係。我們觀察Android執行時的訊息得知Android有取得touch的相關資訊，包含（X，Y）的最大及最小值等等，但不知道在流程中的那個環節發生問題，導致不論觸碰touch的任何位置，Android的畫面總是反應觸碰到（0，0）。Android的source code約 2GB且沒有UML Diagram，因此我們使用暴力法¹⁶及ctags¹⁷來追蹤Android原始碼。我們先從Android執行時的訊息中找到與touch相關的資訊，根據其資訊以frameworks/base/services/java/com/android/server/KeyInputQueue.java為追蹤Android原始碼的切入點。我們發現不論觸碰touch的任何位置，畫面總是反應觸碰到（0，0）的原因是Android application會判斷當下LCD的狀態（ON / OFF）來決定是否處理touch event（LCD關閉時不需要處理touch event），而LCD的狀態又相依於Battery的狀態（電量及是否外接電源等等）。因為我們的實驗平台沒有Battery device及Battery driver，所以Android無法得到Battery的相關資訊，以致Android認為當下的

¹⁵ 可參考 Linux kernel 原始碼 linux/include/linux/input.h。

¹⁶ 需要熟悉 Linux 及 Shell 的操作指令，包含 find、grep、|（pipe）的用法等等。

¹⁷ 我們使用 ctags --C++-kinds=+p -R 來追蹤 Android 原始碼。

LCD是關閉著而忽略touch driver回報的正確(X,Y)，緊接著收到touch driver的SYN Event後，便完成一個完整的touch event，因為(X,Y)初始值為(0,0)，所以Android認為當下的(X,Y)為(0,0)，以致不論我們觸碰touch的任何位置，Android的反應都是(0,0)。因為我們的實驗平台沒有電池裝置，如果要解決touch的問題，必須營造具有電池的假象，於是我們修改程式碼以回報假的Battery狀態並只允許LCD被點亮。我們使用可能造成最少side effects的方式（到目前為止，我們沒有看到任何的side effect），從中截斷BatteryService獲取電源及電池資訊的流程，直接將我們預設的電源狀態、電池狀態及數值寫入BatteryService中，使得Android總是得到良好的電源及電池狀態。接著因為我們的LCM不支援背光調整，所以我們只允許LCD被點亮，因此我們修改hardware/libhardware/power/power.c，並更改相關檔案（Android透過相關檔案來控制硬體的亮暗及背光）的存取權限，使得touch可以正常運作。圖 4-10為經過我們修改之後，在Android環境下，輸入裝置與電源管理的關係，圖 4-11為Android的BatteryService流程，圖 4-12為Android的PowerManagerService流程，圖 4-11及圖 4-12為圖 4-10的細部流程。

相關程式碼的路徑為 linux-2.6.25-android-1.0_r1/drivers/input/touchscreen/ucb1400_ts.c、mydroid/cdma-import/frameworks/base/core/jni/server/com_android_server_BatteryService.cpp、mydroid/cdma-import/hardware/libhardware/power/power.c。

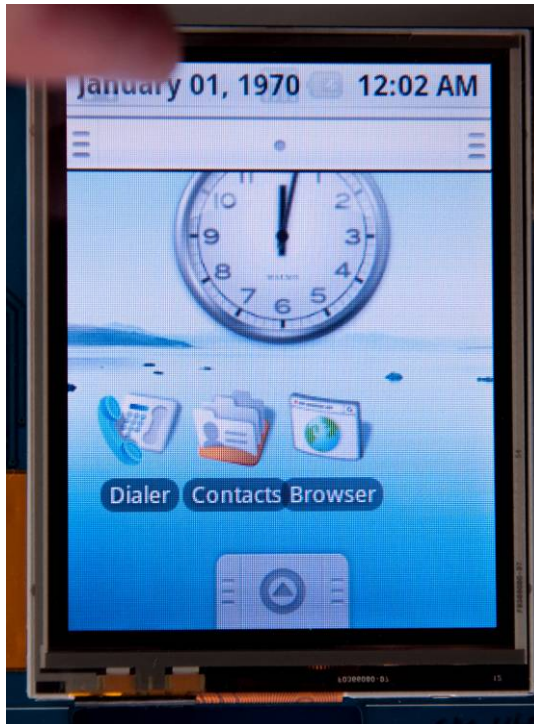


圖 4-8 Android 對 touch 的反應

```
E/EventHub( 1589): could not get driver version for /dev/input/mouse0
I/EventHub( 1589): New device: path=/dev/input/event0 name=android-ke
I/SystemServer( 1589): Starting Bluetooth Service.
I/EventHub( 1589): New keyboard: publicID=65537 device->id=65537 devn
I/SystemServer( 1589): Starting Status Bar Service.
E/EventHub( 1589): could not get driver version for /dev/input/mice,
I/KeyInputQueue( 1589): Device added: id=0x0, name=android-keypad, cl
I/KeyInputQueue( 1589): Device added: id=0x10000, name=null, classes=
I/KeyInputQueue( 1589): X: min=0 max=920 flat=0 fuzz=0
I/KeyInputQueue( 1589): Y: min=0 max=950 flat=0 fuzz=0
I/KeyInputQueue( 1589): Pressure: min=0 max=1 flat=0 fuzz=0
I/KeyInputQueue( 1589): Size: unknown values
I/KeyInputQueue( 1589): absX=com.android.server.InputDevice$AbsoluteI
I/KeyInputQueue( 1589): absY=com.android.server.InputDevice$AbsoluteI
I/KeyInputQueue( 1589): absPressure=com.android.server.InputDevice$Ab
I/KeyInputQueue( 1589): absSize=null
I/foo ( 1589): ***** HAVE TOUCHSCREEN!
I/WindowManager( 1589): Input configuration changed: { scale=1.0 imsi
D/dalvikvm( 1589): GC freed 11328 objects / 708880 bytes in 162ms
I/SystemServer( 1589): Starting Hardware Service.
I/SystemServer( 1589): Starting NetStat Service.
I/SystemServer( 1589): Starting Connectivity Service.
```

圖 4-9 Android 在實驗平台 PXA270 上執行時所顯示的訊息

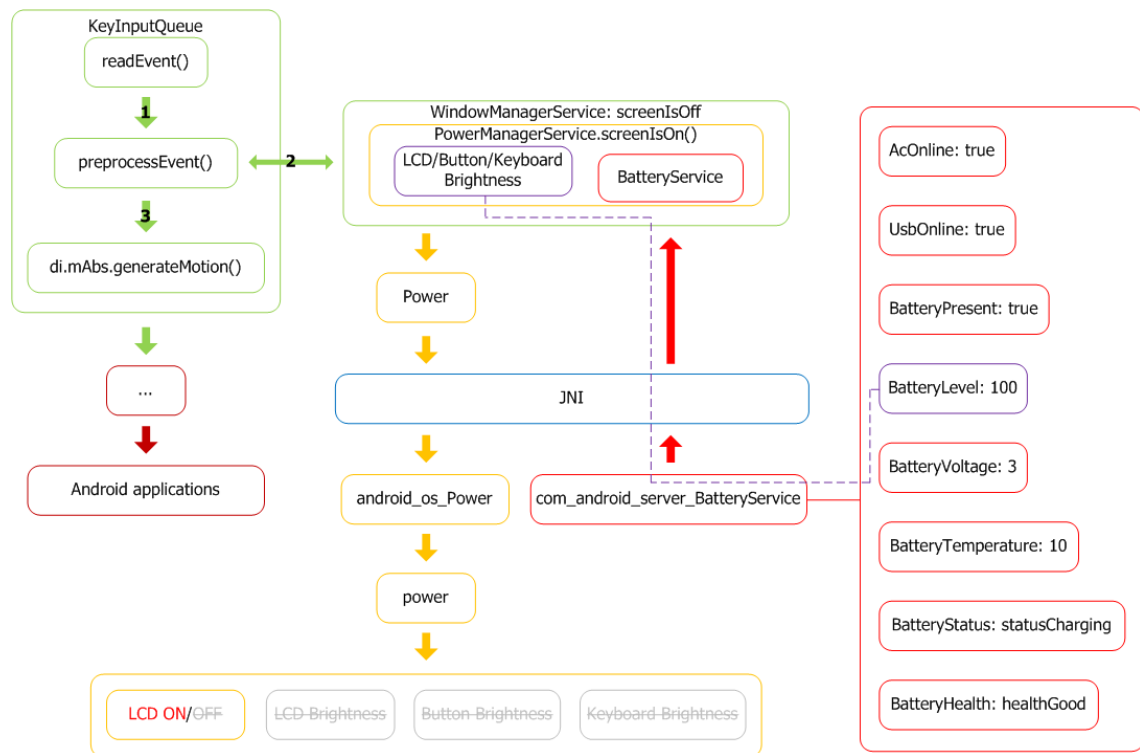


圖 4-10 在 Android 環境下，輸入裝置與電源管理的關係

圖 4-11 為 Android 的 BatteryService 流程。在 Android 的電源管理設計中，對於電源及電池的部份，user space 使用下列介面與 Linux device driver 溝通，並將當下電源及電池的相關狀態記錄於 com_android_server_BatteryService 中：

```

/sys/class/power_supply/ac/online
/sys/class/power_supply/usb/online
/sys/class/power_supply/battery/status
/sys/class/power_supply/battery/health
/sys/class/power_supply/battery/present
/sys/class/power_supply/battery/capacity
/sys/class/power_supply/battery/batt_vol
/sys/class/power_supply/battery/batt_temp
/sys/class/power_supply/battery/technology

```

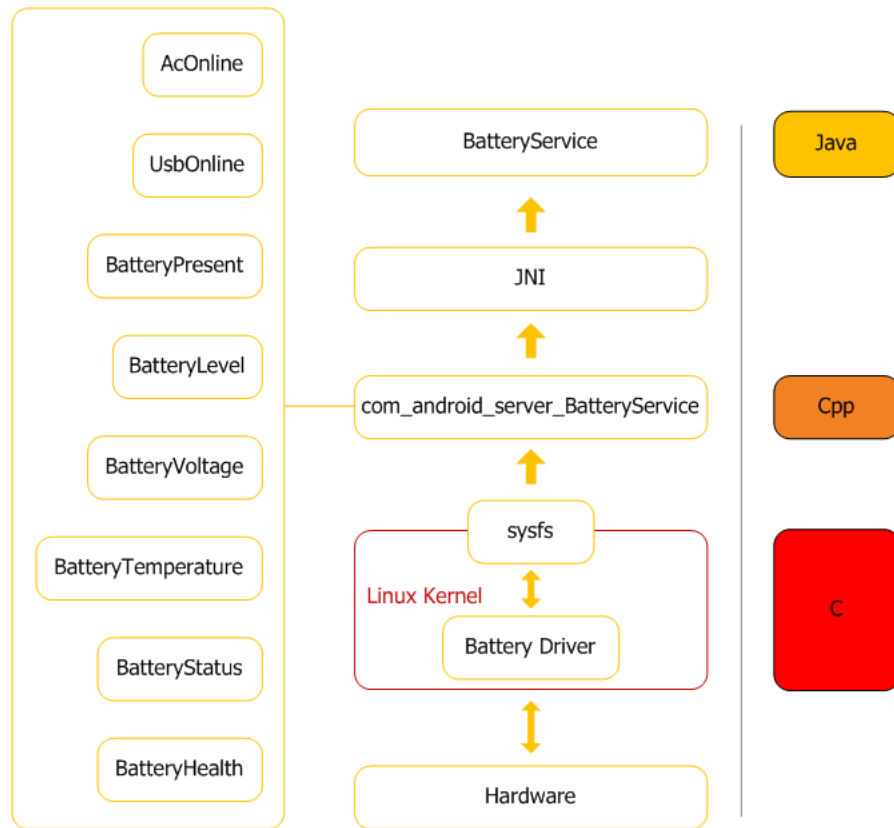


圖 4-11 Android 的 BatteryService 流程

圖 4-12為Android的PowerManagerService流程。Android針對Linux kernel加入電源管理的部份，其相關原始碼位於linux-2.6.25-android-1.0_r1/drivers/android下，user space使用下列介面與Linux device driver溝通而達到電源管理的目的：

```
/sys/android_power/acquire_partial_wake_lock
/sys/android_power/acquire_full_wake_lock
/sys/android_power/release_wake_lock
/sys/android_power/request_state
```

此外，Android 透過下列相關介面來控制硬體裝置的背光程度（相關的 Linux device driver 必須提供相對的支援才能夠達到調整背光的目的）：

```
/sys/class/leds/lcd-backlight/brightness
/sys/class/leds/button-backlight/brightness
/sys/class/leds/keyboard-backlight/brightness
```

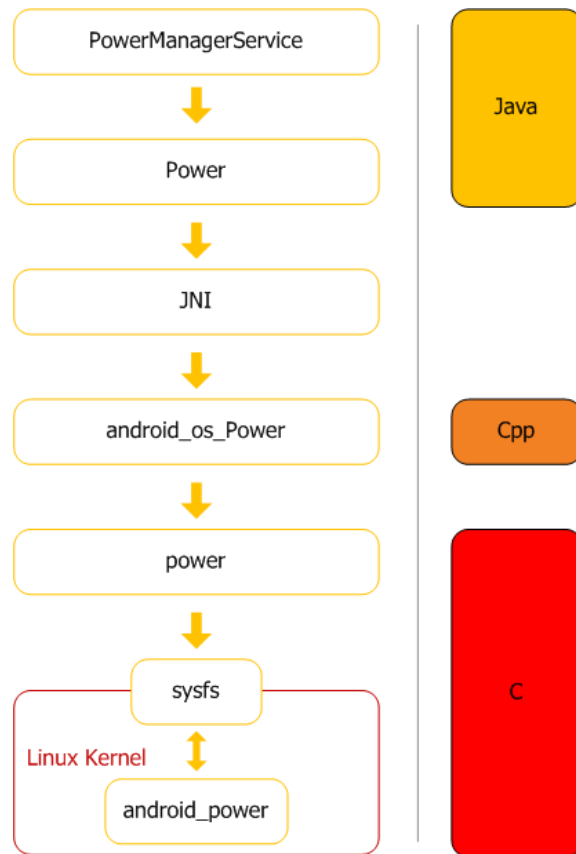


圖 4-12 Android 的 PowerManagerService 流程

4.4.4 編譯 Android

在完成整合性修改之後，我們需要重新編譯Linux kernel及Android，編譯Linux kernel可以參考4.3.3，而Android的部份，Android不支援JDK[29] 6[24]，我們需要先下載JDK 5 (http://java.sun.com/javase/downloads/index_jdk5.jsp)，接著解開JDK，將解開後的bin路徑加入環境變數PATH中即可編譯Android。Android編譯完成後，我們需要取用system及data這兩個目錄來執行Android，這部份可以參考4.6

4.4.5 init.rc

因為我們的實驗平台不是G1 等HTC手機，所以不管是MTD[30] partition的規劃、/system及/data的掛載位置可能皆與Android原先的設置不同，再者為了使touch可以正常運作，我們必須修改部份檔案的存取權限，最後因為我們的實驗平台沒

有藍芽裝置，所以我們將部份與藍芽相關的設置移除，init.rc（附錄F）在經過以上修改之後，Android才能夠在我們的實驗平台上順利地執行。

4.5 所有修改及新增的程式碼列表¹⁸

本論文強調的是移植流程，部份平台相依的程式碼因為平台不同而有所差異，例如 Qualcomm 平台與我們實驗平台所須修改的程式碼就不盡相同，且礙於文章篇幅，我們不提供部份平台相依的程式碼，在此僅列出檔案名稱及路徑。

我們列出在移植過程中所有修改過的檔案列表，灰色斜體字表示在開發過程中為了除錯而修改的檔案，但與最後的移植成果無關。在此我們將程式碼列表分為 Linux kernel / device driver 及 Android 兩個部份。

4.5.1 Linux kernel / device driver

下列為移植 Android 的過程中，針對 Linux kernel / device driver 所修改的檔案名稱及路徑：

linux-2.6.25-android-1.0_r1/kernel/printk.c
linux-2.6.25-android-1.0_r1/kernel/irq/autoprobe.c
linux-2.6.25-android-1.0_r1/init/main.c
linux-2.6.25-android-1.0_r1/drivers/serial/pxa.c
linux-2.6.25-android-1.0_r1/drivers/android/binder.c
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/pxa27x.c
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/generic.h
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/generic.c
linux-2.6.25-android-1.0_r1/arch/arm/kernel/setup.c
linux-2.6.25-android-1.0_r1/Makefile（附錄G）
linux-2.6.25-android-1.0_r1/arch/arm/Makefile（附錄H）
linux-2.6.25-android-1.0_r1/arch/arm/configs/android_pxa270_defconfig（附錄I）
linux-2.6.25-android-1.0_r1/arch/arm/kernel/head.S（附錄J）
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/clock.c（附錄K）
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/irq.c（附錄L）
linux-2.6.25-android-1.0_r1/drivers/android/binder.c

¹⁸ 我們使用（svn）diff 列出所有修改過的檔案

linux-2.6.25-android-1.0_r1/drivers/cpufreq/Kconfig (附錄M)
linux-2.6.25-android-1.0_r1/drivers/i2c/chips/Makefile (附錄N)
linux-2.6.25-android-1.0_r1/drivers/i2c/chips/rtc8564.c¹⁹
linux-2.6.25-android-1.0_r1/drivers/i2c/chips/rtc8564.h¹⁹
linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Kconfig (附錄O)
linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Makefile (附錄P)
linux-2.6.25-android-1.0_r1/drivers/input/touchscreen/ucb1400_ts.c (附錄E)
linux-2.6.25-android-1.0_r1/drivers/net/smc91x.c
linux-2.6.25-android-1.0_r1/drivers/video/pxafb.c (附錄B)
linux-2.6.25-android-1.0_r1/drivers/video/pxafb.h (附錄C)
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/pxa-regs.h
linux-2.6.25-android-1.0_r1/include/linux/config.h (附錄Q)
linux-2.6.25-android-1.0_r1/arch/arm/boot/compressed/Makefile
linux-2.6.25-android-1.0_r1/arch/arm/boot/compressed/head-creator-pxa270.S
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/Kconfig
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/Makefile
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/creator-pxa270-core.c
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/creator-pxa270-irq.c
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/mach-creator-pxa270.c
linux-2.6.25-android-1.0_r1/arch/arm/tools/mach-types
linux-2.6.25-android-1.0_r1/drivers/i2c/chips/Kconfig
linux-2.6.25-android-1.0_r1/drivers/ide/Kconfig
linux-2.6.25-android-1.0_r1/drivers/ide/arm/Makefile
linux-2.6.25-android-1.0_r1/drivers/mtd/chips/jedec_probe.c
linux-2.6.25-android-1.0_r1/drivers/mtd/maps/Kconfig
linux-2.6.25-android-1.0_r1/drivers/mtd/maps/Makefile
linux-2.6.25-android-1.0_r1/drivers/mtd/maps/creator-pxa270-flash.c
linux-2.6.25-android-1.0_r1/drivers/pcmcia/Makefile
linux-2.6.25-android-1.0_r1/drivers/pcmcia/pxa2xx-creator-pxa270.c
linux-2.6.25-android-1.0_r1/drivers/video/Kconfig
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/creator-pxa270.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/creator-regs.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/hardware.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/irqs.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/creator_pxa270_addr.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/creator_pxa270_core.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/creator_pxa270_usb.h

¹⁹ 請自行下載 [Driver for system3's EPSON RTC 8564 chip](#)

linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/def.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/genfont8_8.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/Ata.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/Common.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/Hal4ata.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/Hal4d12.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/Hal4sys.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/Isr.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/basictyp.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/chap_9.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/kthread.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/rbc.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/rbccmd.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/syscnfg.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/tpbulk.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/usb-storage.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/usb.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/lib/usb/usb100.h
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/pxa-regs.h
linux-2.6.25-android-1.0_r1/drivers/input/keyboard/android_keypad.c (附錄D)
linux-2.6.25-android-1.0_r1/include/asm-arm/arch-pxa/android_keypad.h

4.5.2 Android

下列為移植 Android 的過程中，除了 Linux kernel / device driver 以外，所修改的檔案名稱及路徑：

mydroid/cdma-import/build/core/api/current.xml
mydroid/cdma-import/frameworks/base/core/java/android/app/Activity.java
mydroid/cdma-import/frameworks/base/core/java/android/app/ActivityThread.java
mydroid/cdma-import/frameworks/base/core/java/android/app/ApplicationContext.java
mydroid/cdma-import/frameworks/base/core/java/android/text/method/touch.java
mydroid/cdma-import/frameworks/base/core/java/android/util/Log.java
mydroid/cdma-import/frameworks/base/core/java/android/view/MotionEvent.java
mydroid/cdma-import/frameworks/base/core/java/android/view/touchDelegate.java
mydroid/cdma-import/frameworks/base/core/java/android/view/View.java
mydroid/cdma-import/frameworks/base/core/java/android/view/ViewGroup.java
mydroid/cdma-import/frameworks/base/core/java/com/android/internal/widget/LinearL

ayoutWithDefaulttouchReceipient.java
mydroid/cdma-import/frameworks/base/core/jni/server/com_android_server_KeyInput
Queue.cpp
mydroid/cdma-import/frameworks/base/libs/surfaceflinger/DisplayHardware/DisplayH
ardwareBase.cpp
mydroid/cdma-import/frameworks/base/libs/surfaceflinger/RFBServer.cpp
mydroid/cdma-import/frameworks/base/libs/ui/EventHub.cpp
mydroid/cdma-import/frameworks/base/services/java/com/android/server/InputDevice.j
ava
mydroid/cdma-import/frameworks/base/services/java/com/android/server/KeyInputQue
ue.java
mydroid/cdma-import/frameworks/base/services/java/com/android/server/PowerManag
erService.java
mydroid/cdma-import/frameworks/base/services/java/com/android/server/WindowMana
gerService.java
mydroid/cdma-import/system/core/init/init.c
mydroid/cdma-import/system/core/sh/input.c
mydroid/cdma-import/build/core/definitions.mk (附錄R)
mydroid/cdma-import/external/sqlite/dist/Android.mk (附錄S)
mydroid/cdma-import/frameworks/base/core/jni/server/com_android_server_BatterySer
vice.cpp (附錄T)
mydroid/cdma-import/hardware/libhardware/power/power.c (附錄U)

4.6 執行Android

在成功編譯Linux kernel及Android之後，我們將kernel image寫入Flash ROM並確認target可以正常開機。因為我們的實驗平台PXA270 僅有 32MB Flash ROM，不足以存放Android的檔案系統，所以我們使用chroot的方式來執行Android。我們必須先在USB隨身碟中建立一個小型且完整的檔案系統（簡單地說就是基本的目錄結構再加上busybox，圖 3-9為一般Embedded Linux System的目錄結構），接著取用Android成功編譯後的system及data兩個目錄，這兩個目錄的預設路徑為out/target/product/generic/system及out/target/product/generic/data，複製這兩個目錄至USB隨身碟中，圖 4-13為Android在我們實驗平台PXA270 上的目錄結構。最後修改init.rc以符合我們實驗平台的環境，這部份可以參考4.4.5。


```

[root@fc9 usb]# ls
bin      default.prop  init          lib      root      sqlite stmt journals tmp
cache   dev          init.goldfish.rc mnt      sbin      sys
data    etc          init.rc       proc     sdcard    system
[root@fc9 usb]#

```

圖 4-13 Android 在我們實驗平台 PXA270 上的目錄結構

執行 Android 的步驟如下：

1. 開機
2. 掛載 USB 隨身碟
3. chroot 至 USB 隨身碟
4. ./init

附錄V為我們建構Android執行環境的步驟及詳細指令，包含使用我們的開發環境來建構Android執行環境、燒錄kernel image及如何執行Android。

若是在執行Android的過程中遇到問題，可以使用logcat[31]及strace[32]來除錯。

4.7 移植成果

我們的實驗平台PXA270 在Android環境下，經過連續 7 天不關機的測試，仍然可以正常操作，表示我們的實驗平台在Android下具備一定的穩定性，證明我們的移植是成功的，但是我們沒有專業的測試人員進行詳細的測試工作，因此也無法保證系統在操作上沒有任何問題。圖 4-14為Android在我們實驗平台PXA270 的操作圖像，照片編排順序為由左至右，從上至下。我們也錄製Android的操作影片，並上傳至Youtube[33]，圖 4-15為Android在我們實驗平台PXA270 的操作影片擷圖。

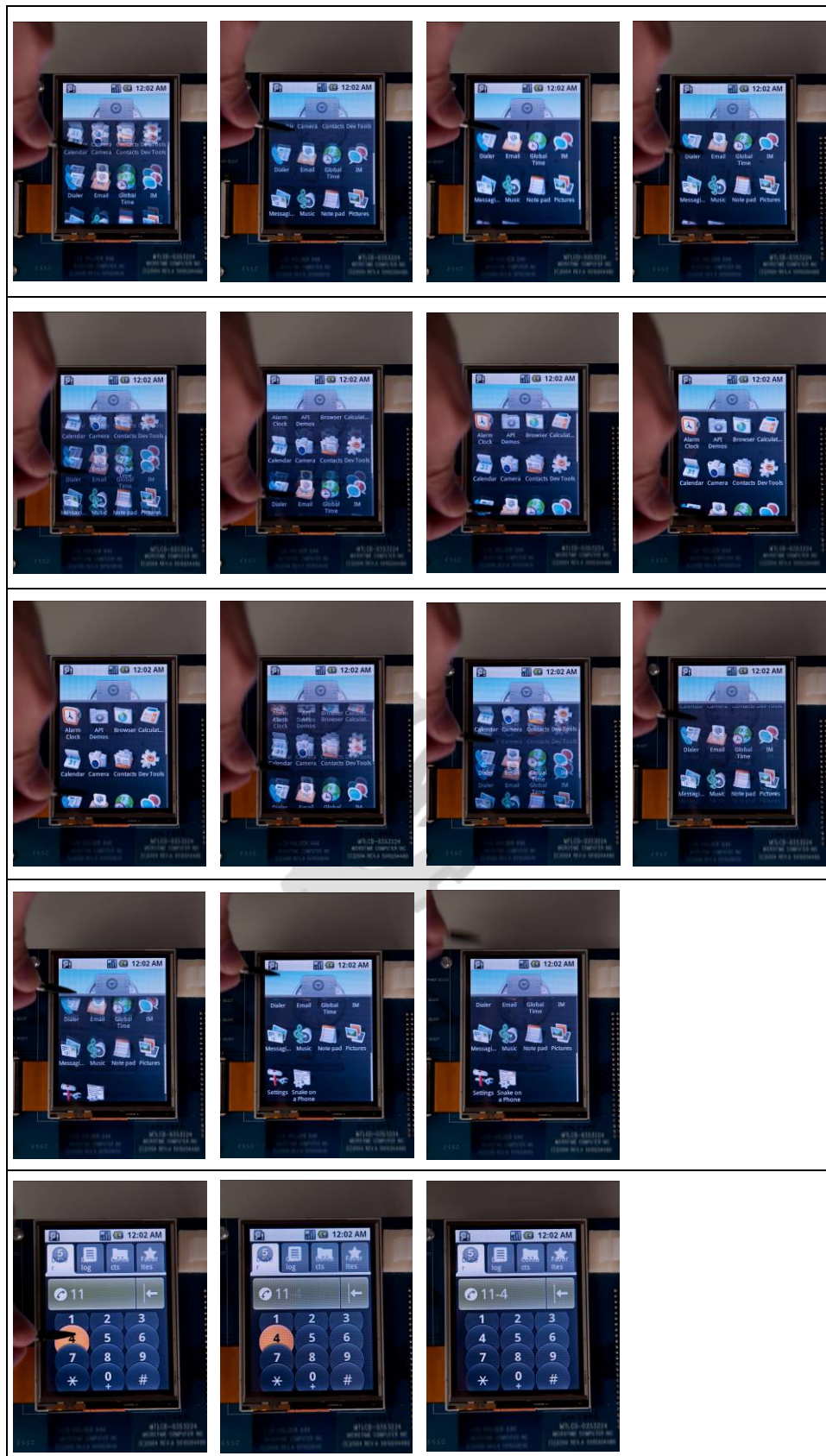


圖 4-14 Android 在我們實驗平台 PXA270 的操作圖像



圖 4-15 Android在我們實驗平台PXA270 的操作影片[34][35][36]

第五章 開發環境簡介

我們在開發過程中深刻感受到 building environment 的重要性，若是有一完善的 building environment，可以加快開發速度、縮短開發時程，達到事半功倍的目的。一個好的 building environment 能夠幫助開發人員處理環境參數及編譯行為等相關前置作業的設定、提供多人共同開發的環境、讓開發人員專注在研發上而不用顧慮因環境不同所衍生的相關問題，所以 building environment 在開發過程中是不可或缺的一環。

要建構一個好的 building environment 有一定的困難度，因為要考慮到實用性、跨平台、平行化編譯、編譯順序、重複編譯等問題，所以一般 building environment 都是為了特定目的而開發，並具備一定客製化的程度。

5.1 一般傳統Building Environment與Android Building Environment簡介

傳統recursive make與Android開發環境的比較如圖 5-1。一般building environment 以Makefile來控制編譯的行為，而Android的開發環境是以Android.mk來取代Makefile的角色。

若以樹狀圖表示開發環境的目錄結構，則一般傳統開發環境除了最末端以外的每個階層中最少會有一個 Makefile，由根目錄的 Makefile 以遞迴方式呼叫下一個階層的 Makefile，透過此方式完成整個編譯程序。此架構的缺點是各階層之間的相依性太強，例如某階層的 Makefile 可能參考到上層甚至是根目錄 Makefile 中的參數等等，造成單一 Makefile 可能影響的縱深非常深，衍生 building environment 日後不易維護且通常無法獨立編譯某階層的某個元件等相關問題。

Android 的開發環境為一整合型的開發環境，它希望此環境下的套件能夠被獨

立地編譯，如此增加平行化編譯的可行性，於是它改寫其中某些套件的編譯方式，希望它們能夠被獨立地編譯且減少相依性。此環境會將編譯時使用到的變數宣告、路徑位置、函數定義全部收集在 build/core 下，並使用各別的 Android.mk 來控制各套件的編譯行為。Android 開發環境的優點是在了解其規則的情況下，開發人員可以很方便地將自己的程式碼加入 Android 中，並在相對應的目錄下加入 Android.mk，而 Android 的開發環境會透過 Android.mk 來編譯相關的程式碼；但缺點是 Android 的開發環境為了達到這種後期開發的便利性，犧牲了部分編譯時間，犧牲的時間是耗費在尋找相關的 Android.mk。為什麼 Android 的開發環境有辦法聰明地找到開發人員新增的 Android.mk，而編譯新增的程式碼？因為在 Android 的開發環境中使用大量 Linux 的 find 指令，因此會對儲存裝置造成較大的負擔，所以開發 Android 要盡可能準備高速或是分散式的儲存裝置，並搭配高階或是多核的 CPU 來加快開發速度。



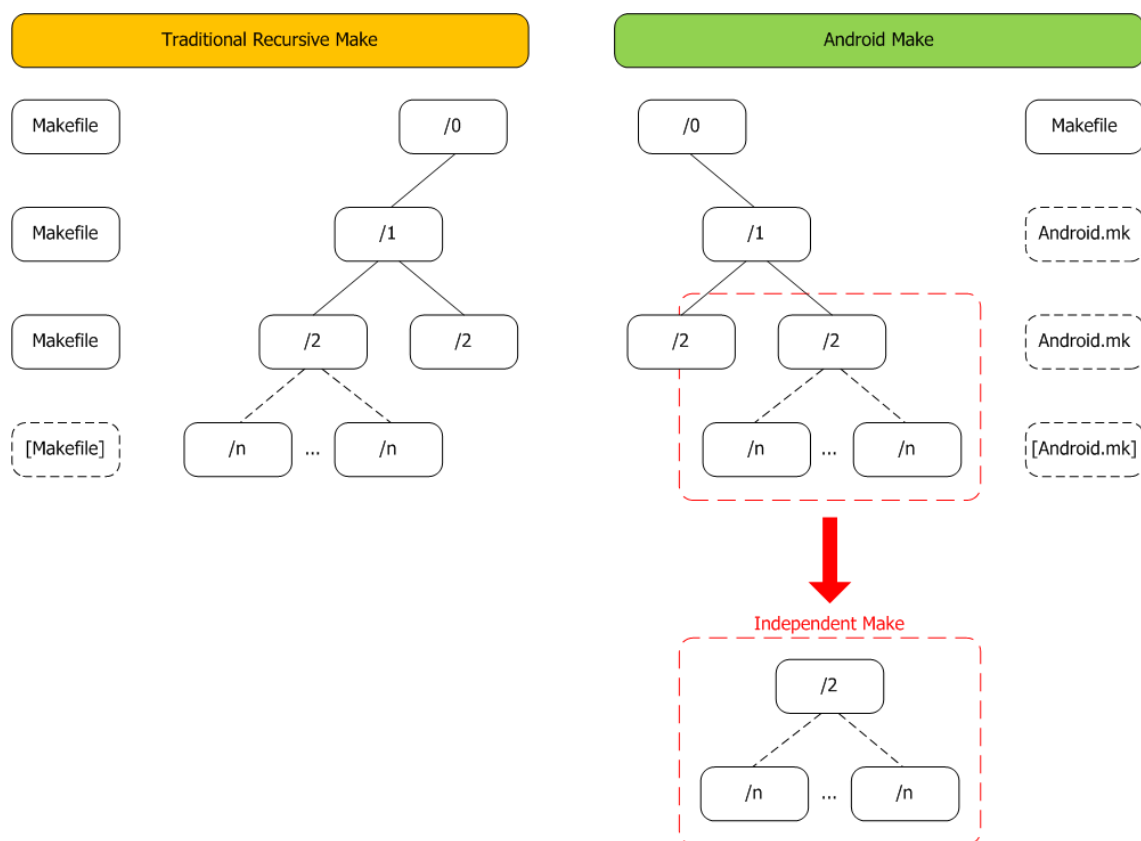


圖 5-1 傳統 recursive make 與 Android 開發環境的比較

若是想了解Android的building environment，卻不知道從何著手時，可以參考GNU Make專案開發工具第三版[37]，因為此書與我們觀察到Android的building environment有部份雷同的概念。

5.2 我們的開發環境簡介

圖 5-2為我們的building environment。雖然建構building environment是一件耗時費工且又無法馬上看到成效的苦差事，但若是沒有一個building environment，我們可能要獨立編譯各個元件，例如Linux kernel、Busybox、root file system等等，而且為了提供多人共同開發的環境、縮短前置作業的時間、讓大家能夠快速地進入Android的世界，所以我們為此打造一個全新、整合性的building environment，我們的building environment不僅小而美，還有下列優點：

1. 分散式且集中管理的 building environment 。
 - 我們將自己建立的 Makefile 收集在 mkfile 目錄下集中管理，避免傳統 recursive make 的 Makefile 散佈在不同階層的目錄中造成管理不易等問題。
 - 單一 Makefile 只負責單一元件，避免傳統單一 Makefile 卻要控制所有元件的編譯行為，造成單一 Makefile 日益增大及維護不易等問題。
 - 降低 Makefile 之間的相依性，building environment 易於維護。
2. 盡可能避免 recursive make，可在 GNU Make 支援 parallel make 的情況下，同時編譯多個目標、縮短編譯時間、加快開發時程。
3. 結構化的 building environment 。
 - Component-based building environment 。
 - 新增的元件可以非常容易地加入我們的 building environment 中。
4. 在 Makefile 中使用函數呼叫取代變數定義，減少變數宣告。
5. 加入判斷式以避免重複編譯等問題。

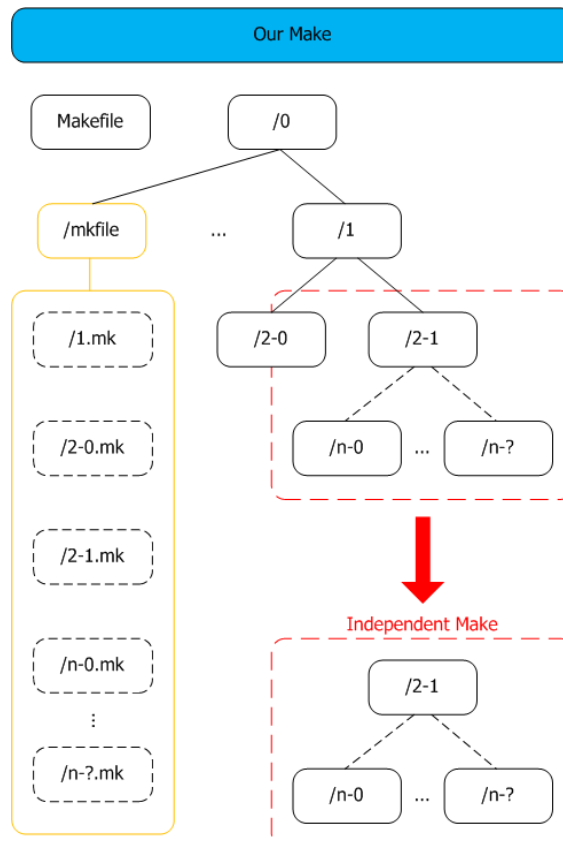


圖 5-2 我們的 building environment

我們設計此開發環境的目的是希望使用者可以簡單透過 make 指令便完成 Android 執行環境的建置，所產生的 Binary Code 可以直接寫入 Flash ROM，並複製 target 端的檔案系統至 USB 隨身碟中，如此實驗平台開機後便可以看到 Android 的執行畫面。

我們的開發環境亦支援以往 Embedded Linux System 的開發方式，我們將 Android 的檔案系統獨立置於 USB 隨身碟中，由使用者自由選擇是否要執行 Android。

5.2.1 版本控制系統

我們使用版本控制系統來管理原始碼，其優點是開發人員可以將心思放在開發工作上，不用擔心程式碼更新、同步、回復、多人協作、存取權限等問題，尤

其在程式碼日益增加的情況下，使用版本控制系統可以不用擔心 Repository（版本控制系統儲存原始碼的檔案空間）的儲存格式、資料庫的設計等相關問題。目前我們的程式碼大小約 3GB，包含 Android：2GB，Linux kernel：0.5GB，toolchain 及其它工具程式：0.5GB，若是沒有使用版本控制系統，實在難以維護如此龐大的程式碼。目前版本控制系統大致分為下列兩類：

1. Centralized：Subversion[38]

2. Distributed：Git[39]

至於其它版本控制系統，我們不在此多作介紹。我們使用 1 作為原始碼的版本控制系統。圖 5-3 為我們開發環境的目錄結構，以下為各目錄的說明：

.config：最上層的設定檔，在執行完 make menuconfig 後自動產生，內容記錄欲編譯的項目及版本，包含 Linux kernel 及其版本、Busybox 及其版本、File System 等等。

LICENSE：開發環境的版權聲明。

Makefile：最上層的 Makefile，用來控制開發環境的編譯行為。

app：target 端的 Application 原始碼，包含 Busybox、Strace 等等。

config：部份軟體套件的設定檔，包含 Linux kernel、Busybox 等設定檔。

doc：開發過程中所參考的相關文件，包含軟硬體等相關文件。

kernel：多個不同版本的 Linux kernel 原始碼。

log：開發過程中的記錄檔，包含開機訊息、除錯訊息、Android 的執行訊息等等。

mkfile：為了使用我們的開發環境編譯各個軟體套件所新增的 Makefile。

mydroid：JDK、Android 原始碼。

rootfs：target 端的檔案系統，包含最陽春、基本的檔案系統、預先編譯好且可以在 PXA270 執行的 Android 檔案系統。

scripts：toolchain、host 端的工具程式，包含 mkimage[40]、mkfs.jffs2[41]、mkyaffs2image[42] 等等。

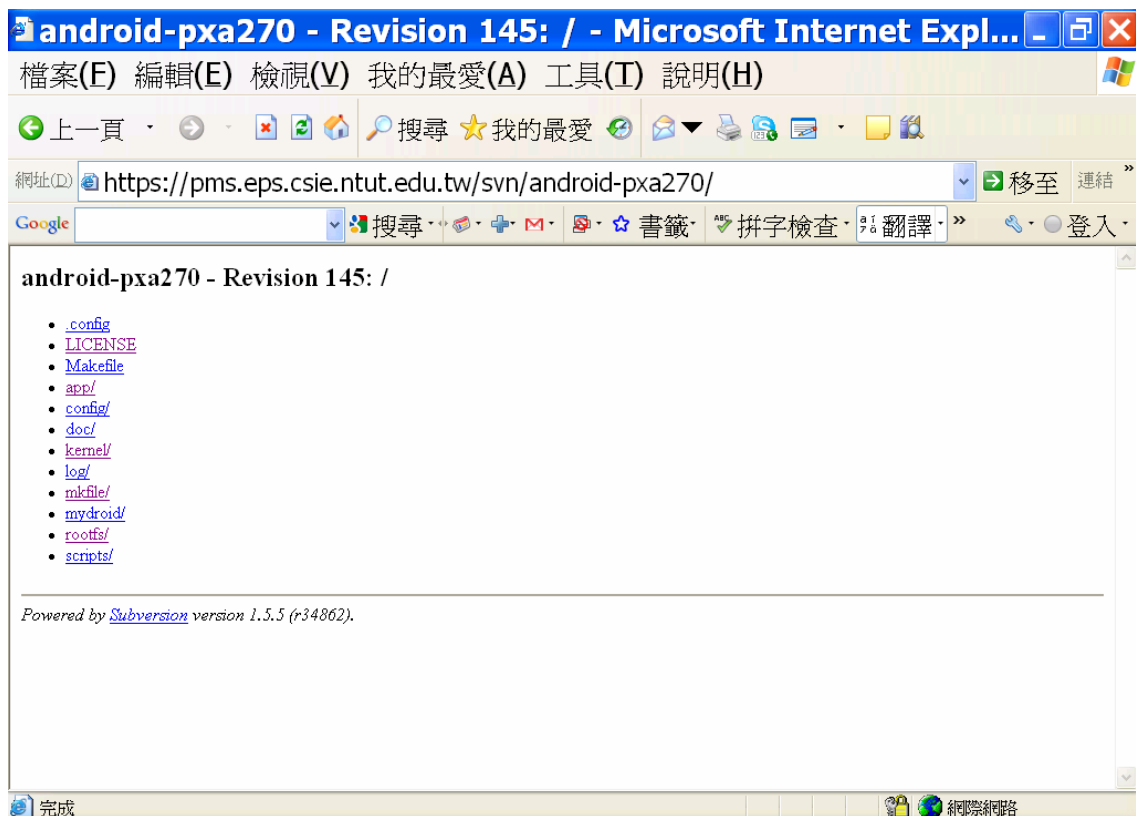


圖 5-3 我們開發環境的目錄結構

5.2.2 如何使用我們的開發環境

我們可以透過make menuconfig來選取欲編譯的項目，如圖 5-4。

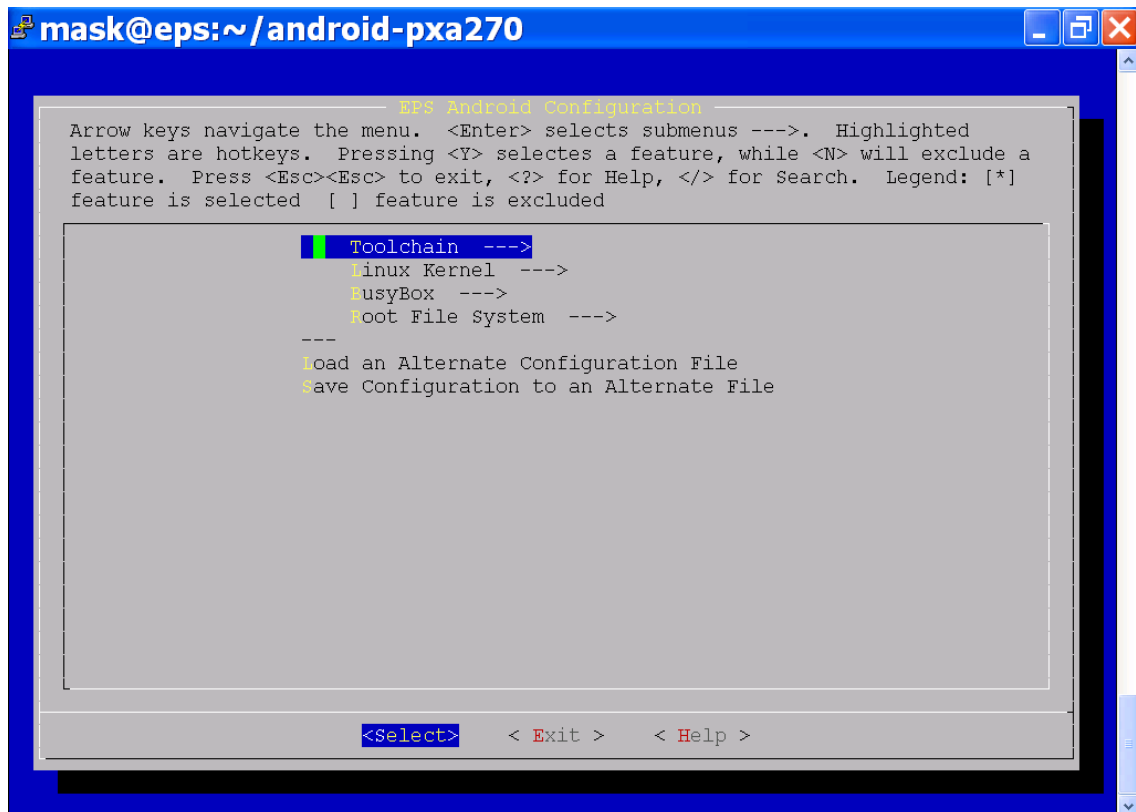


圖 5-4 make menuconfig

圖 5-5為選擇toolchain版本。在我們的開發環境中，toolchain皆是以壓縮檔的形式儲存，我們的開發環境會自動判斷檔案格式（目前支援gzip及bzip2）並進行解壓縮，依據解壓縮的結果自動設定cross compiler，無需使用者額外設定PATH等環境變數，我們的開發環境會幫助使用者處理編譯相關的前置作業。

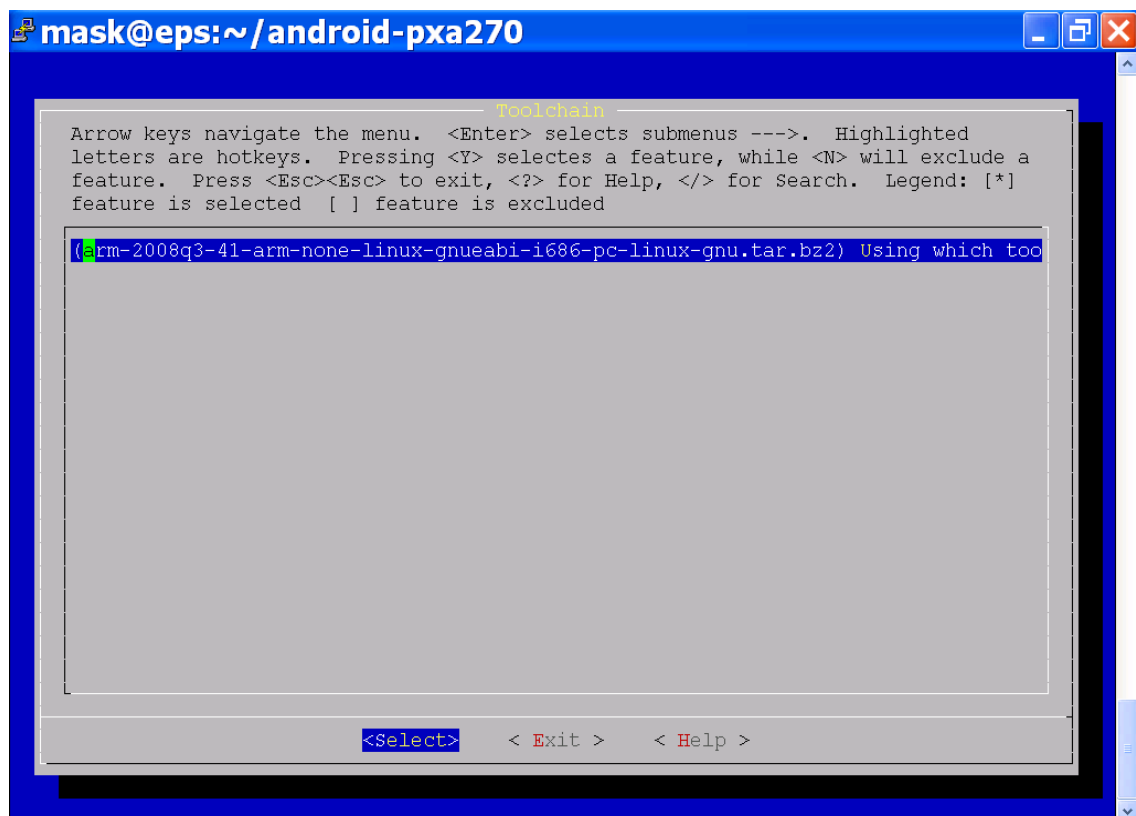


圖 5-5 選擇 toolchain 版本

圖 5-6為選擇Busybox版本。我們提供一般嵌入式系統的開發及執行環境，而Busybox為一般嵌入式系統最常見且最重要的工具之一。



圖 5-6 選擇 Busybox 版本

圖 5-7為選擇Linux kernel版本。我們提供兩個Linux kernel版本：

1. 2.6.15.3（從硬體製造商取得）
2. 2.6.25（我們移植成功的 Linux kernel，包含 Android 的 patch）



圖 5-7 選擇 Linux kernel 版本

圖 5-8為選擇檔案系統，包含Android的版本（我們提供一個預先編譯好且可以在實驗平台PXA270 執行的Android demo file system，預先編譯好的目的是為了節省時間並作為日後開發的基準點，因為編譯Android需要耗費一定的時間成本，相關數據可以參考5.2.3。再者我們提供依PXA270 而修改的完整Android程式碼）、檔案系統的格式等等。

我們提供一個最基本、陽春的檔案系統，大約 1~2MB，方便使用者從事以往 Embedded Linux System 的開發工作，並提供一個預先編譯好且可以在我們實驗平台 PXA270 執行的 Android demo file system，讓使用者自由選擇想要的環境。

我們的開發環境適用於Linux x86 及Linux x86_64，使用者無須擔心編譯時相關參數的設定問題，我們的開發環境會自動選擇正確的JDK來進行編譯程序。因為我們的實驗平台採用NOR Flash，所以我們目前使用jffs2[41]的檔案格式。

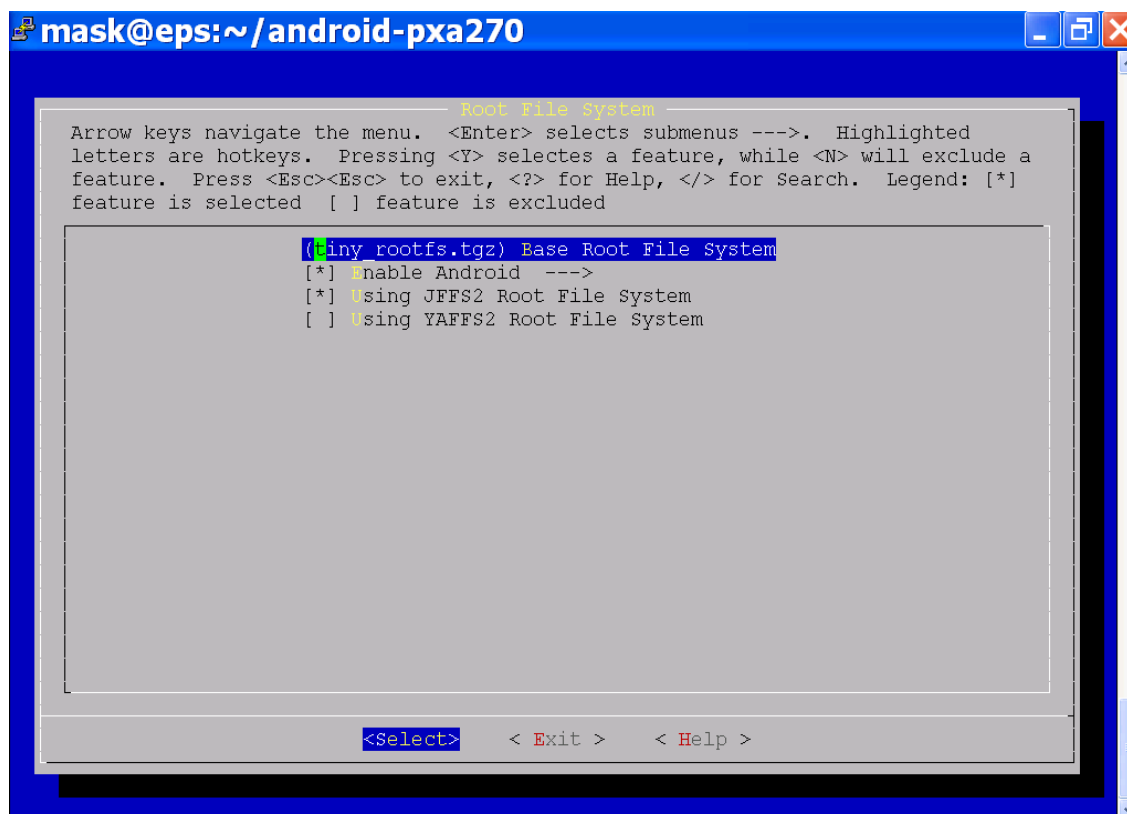


圖 5-8 選擇檔案系統

5.2.3 測試平台

為確保我們建構的開發環境能夠在不同平台上順利地編譯，因此我們分別在幾個不同平台上進行相同的編譯測試，表 5-1 為我們的開發環境在不同平台上的測試數據。

表 5-1 我們的開發環境在不同平台上的測試數據

	CentOS release 5.3 (Final) / Red Hat Enterprise Linux Server release 5 (Tikanga)	Fedora release 7 (Moonshine)	Ubuntu 8.04.2
Linux kernel	2.6.18-128.1.10.el5 , x86_64	2.6.23.17-88.fc7	2.6.24-23-generic
CPU	4x Intel(R) Xeon(TM) MP CPU 3.16GHz	2x Intel(R) Core(TM)2 CPU 6320 @ 1.86GHz	AMD Athlon(tm) 64 Processor 3000+
DRAM	4GB	1GB	2GB
GNU Make Version	3.81	3.81	3.81
make	1 : 28 : 39	1 : 52 : 07	1 : 20 : 20
make -j	53 : 48	FAIL	2 : 47 : 19
make -j2	53 : 51	1 : 20 : 36	N/A
make -j4	47 : 37	N/A	N/A

第六章 結論

Android毋庸置疑已經成為當下最熱門的話題之一，我們可以發現越來越多的品牌大廠紛紛推出Android手機及其它使用Android系統的電子產品，包含已經在市面上販售的HTC手機[43][44]及其它準備上市的電子產品，包含Acer[45]及Samsung[46]也在今年的Computex展及新加坡電信展展出Android的相關產品，由此可見Android的效應才正在發酵，未來想必可以見到更多使用Android系統的電子產品。

Android之所以備受矚目，原因不外乎Google是其最大的支持者，Google不僅整合了以往一般嵌入式系統最缺乏的UI，且免費及開放式的軟體平台讓任何人都可以自由地修改或新增Android的功能，這也是Android可以在短時間內掀起這麼大波瀾的原因。

本論文以移植的流程為主軸，使用引導的方式，由淺入深、從大方向到小細節、從概念、流程到程式碼，盡可能詳述Android移植的過程，且修改程式碼的原則盡可能以可讀性及最少的修改來達到我們預期的目的。從系統整合的角度來看，要完成Android系統整合的工作並非易事，更何況我們是在毫無硬體廠商支援下獨立完成開發工作，所以我們希望公開移植的流程，分享移植的經驗，在不造成侵權的前提下，盡可能公開移植的程式碼，這也正是Open Source及Android的理念。

目前我們已經完成實驗平台PXA270的LCD、Keypad、Touch、USB、有線網路等Android的移植工作，尚未完成Audio、無線網路及部份周邊硬體裝置的Android移植工作，這也是我們正在努力的目標。

參考文獻

- [1] Linux Online - About the Linux Operating System, <http://www.linux.org/info/>
- [2] Android | Official Website, <http://www.android.com/>
- [3] The new T-Mobile G1 with Google cell phone – Official Site,
<http://www.t-mobileg1.com/>
- [4] HTC - Products - HTC Dream – Overview,
<http://www.htc.com/www/product/dream/overview.html>
- [5] Canonical developers aim to make Android apps run on Ubuntu - Ars Technica,
<http://arstechnica.com/open-source/news/2009/05/canonical-developers-aim-to-make-android-apps-run-on-ubuntu.ars>
- [6] Android port to MIPS completed - News - Linux for Devices ,
<http://www.linuxfordevices.com/c/a/News/Android-port-to-MIPS-completed/>
- [7] Intel® PXA270 Processor for Embedded Computing – Overview,
<http://www.intel.com/design/embeddedpca/applicationsprocessors/302302.htm>
- [8] Welcome (Android Open Source Project), <http://source.android.com/>
- [9] The GNU Operating System, <http://www.gnu.org/>
- [10] GNU Make - GNU Project - Free Software Foundation (FSF),
<http://www.gnu.org/software/make/>
- [11] Android Emulator | Android Developers,
<http://developer.android.com/guide/developing/tools/emulator.html>
- [12] What is Android? | Android Developers,
<http://developer.android.com/guide/basics/what-is-android.html>
- [13] BSD licenses - Wikipedia, the free encyclopedia,
http://en.wikipedia.org/wiki/BSD_licenses

- [14] Developing In Eclipse, with ADT | Android Developers,
<http://developer.android.com/guide/developing/eclipse-adt.html>
- [15] Eclipse.org home, <http://www.eclipse.org/>
- [16] Android 1.5 SDK, Release 2 | Android Developers,
http://developer.android.com/sdk/1.5_r2/index.html
- [17] Qualcomm Home, <http://www.qualcomm.com/>
- [18] Welcome to LiMo, <http://www.limofoundation.org/>
- [19] Open Handset Alliance, <http://www.openhandsetalliance.com/>
- [20] blob, a StrongARM boot loader | Get blob, a StrongARM boot loader at
SourceForge.net, <http://sourceforge.net/projects/blob/>
- [21] Das U-Boot - Universal Bootloader | Get Das U-Boot - Universal Bootloader at
SourceForge.net, <http://sourceforge.net/projects/u-boot>
- [22] BusyBox, <http://www.busybox.net/>
- [23] ARM - The Architecture for the Digital World, <http://www.arm.com/>
- [24] Get source (Android Open Source Project), <http://source.android.com/download>
- [25] WinMerge, <http://winmerge.org/>
- [26] Meld : Home Page, <http://meld.sourceforge.net/>
- [27] Exuberant Ctags, <http://ctags.sourceforge.net/>
- [28] Linux NFS faq, <http://nfs.sourceforge.net/>
- [29] Java Development Kit - Wikipedia, the free encyclopedia,
http://en.wikipedia.org/wiki/Java_Development_Kit
- [30] Memory Technology Device (MTD) Subsystem for Linux.,
<http://www.linux-mtd.infradead.org/>
- [31] Android Debug Bridge | Android Developers,
<http://developer.android.com/guide/developing/tools/adb.html>

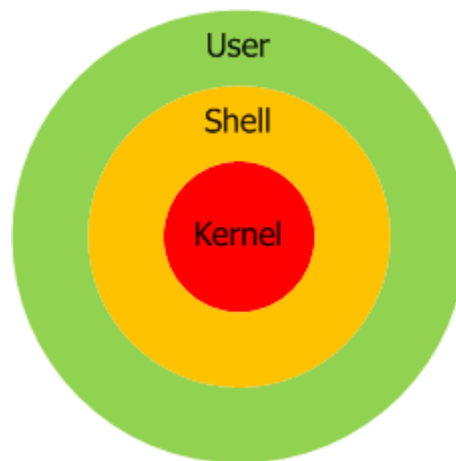
- [32] strace | Get strace at SourceForge.net, <http://sourceforge.net/projects/strace/>
- [33] YouTube - Broadcast Yourself ◦ , <http://www.youtube.com/>
- [34] YouTube - Android touch works smoothly on PXA270 (date: 20090515),
<http://www.youtube.com/watch?v=IYzRSNuUsIw>
- [35] YouTube - Android keypad works fine on PXA270 (date: 20090415),
<http://www.youtube.com/watch?v=3Ine1k4Hzy8>
- [36] YouTube - Android browser on PXA270,
<http://www.youtube.com/watch?v=CNGhFmCwaxw>
- [37] Welcome to O'Reilly Taiwan!,
http://www.oreilly.com.tw/product_unix.php?id=a164
- [38] subversion.tigris.org, <http://subversion.tigris.org/>
- [39] Git - Fast Version Control System, <http://git-scm.com/>
- [40] WebHome < U-Boot < DENX, <http://www.denx.de/wiki/U-Boot>
- [41] JFFS2: The Journalling Flash File System, version 2,
<http://sources.redhat.com/jffs2/>
- [42] YAFFS | A Flash file system for embedded use, <http://www.yaffs.net/>
- [43] HTC - Products - HTC Magic – Overview,
<http://www.htc.com/tw/product/magic/overview.html>
- [44] HTC, <http://www.htc.com>
- [45] [Computex 2009] 第一天的意外驚喜，Acer Android小筆電動手玩，
<http://taiwan.cnet.com/digilife/0,2000089053,20138618-20001421c,00.htm>
- [46] Samsung I7500，Samusng第一款Android手機正式發表，
<http://taiwan.cnet.com/crave/0,2000088746,20137861,00.htm>

附錄A Shell 簡介

什麼是 Shell？簡單的說，Shell 是一軟體，介於 Linux kernel 與使用者之間，使用者透過 Shell 與 Linux kernel 溝通。

Shell 提供一操作環境，方便使用者輸入某些特定指令，並將指令執行結果顯示於螢幕上。

附錄-圖 1 為一般Linux系統的軟體架構。介於Shell與Linux kernel之間當然還存在著許多Library及Application，但附錄-圖 1 僅是為了簡單表現出Shell在Linux系統中所扮演的角色及所在的位置。



附錄-圖 1 一般 Linux 系統的軟體架構

附錄B Framebuffer driver (pxafb.c)

```
--- kernel.git/drivers/video/pxafb.c      2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/drivers/video/pxafb.c    2009-06-12 10:36:05.
000000000 +0800
@@ -70,6 +70,18 @@
    static char g_options[PXAFB_OPTIONS_SIZE] __devinitdata = "";
    #endif

+static int pxa_fb_pan_display(struct fb_var_screeninfo *var, struct fb_info *    info)
+{
+    struct pxa_fb_info *fbi = (struct pxa_fb_info *)info;
+    unsigned long flags;
+
+    local_irq_save(flags);
+
+    fbi->task_state = C_ANDROID_NOP;
+    local_irq_restore(flags);
+    return 0;
+}
+
static inline void pxa_fb_schedule_work(struct pxa_fb_info *fbi, u_int state)
{
    unsigned long flags;
@@ -271,7 +283,7 @@
    var->sync                = mode->sync;
    var->grayscale            = mode->cmap_grayscale;
    var->xres_virtual         = var->xres;
-    var->yres_virtual         = var->yres;
+    var->yres_virtual         = var->yres * 2;
}

/*
@@ -496,6 +508,7 @@
    .fb_imageblit            = cfb_imageblit,
    .fb_blank                = pxa_fb_blank,
    .fb_mmap                 = pxa_fb_mmap,
```

```

+         .fb_pan_display = pxafb_pan_display,
+     };

+     /*
@@ -688,13 +701,13 @@

+         /* populate descriptors */
+         fbi->dmadesc_fblow_cpu->fdadr = fbi->dmadesc_fblow_dma;
-         fbi->dmadesc_fblow_cpu->fsadr = fbi->screen_dma + BYTES_PER_PANEL;
+         fbi->dmadesc_fblow_cpu->fsadr = fbi->screen_dma + BYTES_PER_PANEL
+         ((fbi->fb.var.yoffset / fbi->fb.var.yres) * (fbi->fb.var.yres) * (fbi->fb.var.xres) *
2);

+         fbi->dmadesc_fblow_cpu->fidr = 0;
+         fbi->dmadesc_fblow_cpu->ldcmd = BYTES_PER_PANEL;

+         fbi->fdadr1 = fbi->dmadesc_fblow_dma; /* only used in dual-panel mode */

-         fbi->dmadesc_fbhigh_cpu->fsadr = fbi->screen_dma;
+         fbi->dmadesc_fbhigh_cpu->fsadr = fbi->screen_dma + ((fbi->fb.var.yoffset /
fbi->fb.var.yres) * (fbi->fb.var.yres) * (fbi->fb.var.xres) * 2);
+         fbi->dmadesc_fbhigh_cpu->fidr = 0;
+         fbi->dmadesc_fbhigh_cpu->ldcmd = BYTES_PER_PANEL;

@@ -1166,14 +1179,14 @@

+         fbi->fb.fix.type = FB_TYPE_PACKED_PIXELS;
+         fbi->fb.fix.type_aux = 0;
+         fbi->fb.fix.xpanstep = 0;
-         fbi->fb.fix.ypanstep = 0;
+         fbi->fb.fix.ypanstep = 1;
+         fbi->fb.fix.ywrapstep = 0;
+         fbi->fb.fix.accel = FB_ACCEL_NONE;

+         fbi->fb.var.nonstd = 0;
+         fbi->fb.var.activate = FB_ACTIVATE_NOW;
-         fbi->fb.var.height = -1;
-         fbi->fb.var.width = -1;
+         fbi->fb.var.height = 71;
+         fbi->fb.var.width = 53;

```

```
fbi->fb.var.accel_flags = 0;
fbi->fb.var.vmode      = FB_VMODE_NONINTERLACED;
```

@@ -1198,6 +1211,7 @@

```

+
    for (i = 0; i < inf->num_modes; i++) {
        smemlen = mode[i].xres * mode[i].yres * mode[i].bpp / 8;
        smemlen *= 2;
        if (smemlen > fbi->fb.fix.smem_len)
            fbi->fb.fix.smem_len = smemlen;
    }
```



附錄C pxafb.h

```
diff -raNu kernel.git/drivers/video/pxafb.h linux-2.6.25-android-1.0_r1/drivers/video/pxafb.h
```

```
--- kernel.git/drivers/video/pxafb.h      2008-07-24 09:04:04.000000000 +0800
```

```
+++ linux-2.6.25-android-1.0_r1/drivers/video/pxafb.h    2009-05-09
```

```
18:54:47.000000000 +0800
```

```
@@ -109,6 +109,7 @@
```

```
#define C_DISABLE_PM                (5)
```

```
#define C_ENABLE_PM                 (6)
```

```
#define C_STARTUP                    (7)
```

```
+#define C_ANDROID_NOP              (99)
```

```
#define PXA_NAME                    "PXA"
```



附錄D Keypad driver (android_keypad.c)

```
/*
 * linux/drivers/input/keyboard/android_keypad.c
 *
 * Modified from driver for the pxa27x matrix keyboard controller.
 * Modified from linux/drivers/input/keyboard/pxa27x_keypad.c
 *
 * Modified: April 6, 2009
 * Author: Mask <cycdisk@gmail.com>
 *
 * Created: Feb 22, 2007
 * Author: Rodolfo Giometti <giometti@linux.it>
 *
 * Based on a previous implementations by Kevin O'Connor
 * <kevin_at_koconnor.net> and Alex Osborne <bobofdoom@gmail.com> and
 * on some suggestions by Nicolas Pitre <nico@cam.org>.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 */

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/interrupt.h>
#include <linux/input.h>
#include <linux/device.h>
#include <linux/platform_device.h>
#include <linux/clock.h>
#include <linux/err.h>
#include <linux/delay.h>

#include <asm/mach-types.h>
#include <asm/mach/arch.h>
```

```

#include <asm/mach/map.h>

#include <asm/arch/hardware.h>
#include <asm/arch/android_keypad.h>

#include <linux/kthread.h>
#include <linux/freezer.h>

#define MAX_MATRIX_KEY_NUM (4 * 4)
#define LOG_LEVEL KERN_ALERT

struct android_keypad {
    struct android_keypad_platform_data *pdata; // maybe take off
    struct input_dev *input_dev;
    unsigned long jiffy;
    unsigned long jiffy_diff;
    unsigned long jiffy_start_polling;
    struct task_struct *polling_thread;
    wait_queue_head_t keypad_wq;
    struct mutex mutex;
    spinlock_t spinlock;
    unsigned long idle_period;
    int polling;
};

static DEFINE_MUTEX(mutex_io_reg2);
void io_reg2_write(unsigned short val)
{
    mutex_lock(&mutex_io_reg2);
    IO_REG2 = val;
    mutex_unlock(&mutex_io_reg2);
}

static DEFINE_MUTEX(mutex_io_reg1);
unsigned short io_reg1_read(void)
{
    unsigned short val;
    mutex_lock(&mutex_io_reg1);

```

```

    val = IO_REG1;
    mutex_unlock(&mutex_io_reg1);
    return val;
}

/* 1(KEY_BACK) 2(KEY_UP) 3(KEY_H) A(KEY_MENU)
 * 4(KEY_LEFT) 5(KEY_RESERVED) 6(KEY_RIGHT) B(KEY_HOME)
 * 7(KEY_T) 8(KEY_DOWN) 9(KEY_P) C(KEY_BACKSPACE)
 * *(KEY_LEFTSHIFT)0(KEY_DOT) #(KEY_SLASH) D(KEY_SPACE)
 */
struct android_keymap {
    unsigned int hw_key;
    unsigned int keycode;
    unsigned int shift_keycode;
};

#define KEY(r3,r2,r1,r0, c3,c2,c1,c0) \
( (r3<<7)|(r2<<6)|(r1<<5)|(r0<<4)|(c3<<3)|(c2<<2)|(c1<<1)|c0 )

static struct android_keymap keymap[] = {
{KEY(1,1,1,0, 1,1,1,0), KEY_1, KEY_BACK},
{KEY(1,1,1,0, 1,1,0,1), KEY_2, KEY_UP},
{KEY(1,1,1,0, 1,0,1,1), KEY_3, KEY_H},
{KEY(1,1,1,0, 0,1,1,1), KEY_A, KEY_MENU},
{KEY(1,1,0,1, 1,1,1,0), KEY_4, KEY_LEFT},
{KEY(1,1,0,1, 1,1,0,1), KEY_5, KEY_RESERVED},
{KEY(1,1,0,1, 1,0,1,1), KEY_6, KEY_RIGHT},
{KEY(1,1,0,1, 0,1,1,1), KEY_B, KEY_HOME},
{KEY(1,0,1,1, 1,1,1,0), KEY_7, KEY_T},
{KEY(1,0,1,1, 1,1,0,1), KEY_8, KEY_DOWN},
{KEY(1,0,1,1, 1,0,1,1), KEY_9, KEY_P},
{KEY(1,0,1,1, 0,1,1,1), KEY_C, KEY_BACKSPACE},
{KEY(0,1,1,1, 1,1,1,0), KEY_LEFTSHIFT, KEY_LEFTSHIFT},
{KEY(0,1,1,1, 1,1,0,1), KEY_0, KEY_DOT},
{KEY(0,1,1,1, 1,0,1,1), KEY_0, KEY_SLASH},
{KEY(0,1,1,1, 0,1,1,1), KEY_D, KEY_SPACE},
{0,KEY_UNKNOWN, KEY_UNKNOWN },
};

static void android_keypad_setkeycode(struct android_keypad *keypad)

```

```

{
    struct input_dev *input_dev = keypad->input_dev;
    unsigned int i;
    for(i = 0; keymap[i].keycode != KEY_UNKNOWN; ++i)
    {
        set_bit(keymap[i].keycode,input_dev->keybit);
        set_bit(keymap[i].shift_keycode,input_dev->keybit);
    }
}

```

```

static unsigned int lookup_keycode(unsigned int hw_key)
{
    unsigned int i;
    static bool shift_key = false; // maybe change to static
    for(i = 0; keymap[i].keycode != KEY_UNKNOWN; ++i)
    {
        if(keymap[i].hw_key == hw_key)
            break;
    }
    if(keymap[i].keycode == KEY_LEFTSHIFT)
        shift_key = !shift_key;
    if(shift_key)
        return keymap[i].shift_keycode;
    return keymap[i].keycode;
}

```

```

#define MAX_IDLE_MSEC (1*700) // 0.7 sec
#define MIN_IDLE_MSEC (1*100) // 0.1 sec
#define IDLE_STEPS 3
static unsigned int min_idle_msec = MIN_IDLE_MSEC;
static unsigned int max_idle_msec = MAX_IDLE_MSEC;
static unsigned int idle_steps = IDLE_STEPS;
#define SCAN(r3,r2,r1,r0) ( (r3<<3)|(r2<<2)|(r1<<1)|r0 )
unsigned short scan_key[] = {
    SCAN(1,1,1,0),
    SCAN(1,1,0,1),
    SCAN(1,0,1,1),
    SCAN(0,1,1,1),

```

```

};
static int android_keypad_thread(void * data)
{
    struct android_keypad *keypad = data;
    struct sched_param param = { .sched_priority = 1 };
    unsigned short scan;
    unsigned int keycode;
    int i;

    keypad->idle_period = min_idle_msec;
    sched_setscheduler(current, SCHED_FIFO, &param);
    current->flags |= PF_NOFREEZE;
    do {
        i = 0;
        do
        {
            scan = scan_key[i];
            io_reg2_write(scan); // maybe change to 0xe because bit 4~7 don't care
            keycode = lookup_keycode( (scan << 4) | ((io_reg1_read() & 0x0f00) >>
8));

            if(keycode != KEY_UNKNOWN)
                break;
            ++i;
        } while(i < ARRAY_SIZE(scan_key));
        if(keycode == KEY_UNKNOWN) // no input --- need to modify
        {
            if(keypad->idle_period < max_idle_msec)
                keypad->idle_period += (max_idle_msec -
min_idle_msec)/idle_steps;
            if(keypad->idle_period > max_idle_msec)
                keypad->idle_period = max_idle_msec;
        }
        else
        {
            // calculate jiffies to cancel fast continuing key
            keypad->jiffy_diff = jiffies - keypad->jiffy;
            keypad->jiffy = jiffies;
            input_report_key(keypad->input_dev, keycode, 1);

```

```

        input_sync(keypad->input_dev);
        input_report_key(keypad->input_dev,keycode,0);
        input_sync(keypad->input_dev);
        keypad->idle_period = min_idle_msec;
        printk(LOG_LEVEL "%s: jiffy_diff = %lu\n",
__func__,keypad->jiffy_diff);
        printk(LOG_LEVEL "%s: keycode = %d\n",__func__,keycode);
    }
    set_task_state(current, TASK_INTERRUPTIBLE);
    if (!kthread_should_stop())
        schedule_timeout(msecs_to_jiffies(keypad->idle_period));
    set_task_state(current, TASK_RUNNING);
} while(!kthread_should_stop());
return 0;
}

static int android_keypad_open(struct input_dev *dev)
{
    struct android_keypad *keypad = input_get_drvdata(dev); // remember check
"private" in _probe function
    int err;

    keypad->polling_thread =
kthread_run(android_keypad_thread,keypad,"kandroid_keypadd");
    if(IS_ERR(keypad->polling_thread))
    {
        err = PTR_ERR(keypad->polling_thread);
        printk(LOG_LEVEL "%s: create kthread ERROR: %d\n",__func__,err);
        return err;
    }

    return 0;
}

static void android_keypad_close(struct input_dev *dev)
{
    struct android_keypad *keypad = input_get_drvdata(dev);

```

```

        kthread_stop(keypad->polling_thread);
    }

#ifdef CONFIG_PM
static int android_keypad_suspend(struct platform_device *pdev, pm_message_t state)
{
    struct android_keypad *keypad = platform_get_drvdata(pdev);

    mutex_lock(&keypad->mutex);
    keypad->idle_period = max_idle_msec;
    mutex_unlock(&keypad->mutex);

    return 0;
}

static int android_keypad_resume(struct platform_device *pdev)
{
    struct android_keypad *keypad = platform_get_drvdata(pdev);

    mutex_lock(&keypad->mutex);
    keypad->idle_period = min_idle_msec;
    mutex_unlock(&keypad->mutex);

    return 0;
}
#else
#define android_keypad_suspend NULL
#define android_keypad_resume NULL
#endif

static int __devinit android_keypad_probe(struct platform_device *pdev)
{
    struct android_keypad *keypad;
    struct input_dev *input_dev;
    int error;

    keypad = kzalloc(sizeof(struct android_keypad), GFP_KERNEL);
    if (keypad == NULL) {

```



```

        dev_err(&pdev->dev, "failed to allocate driver data\n");
        return -ENOMEM;
    }

    mutex_init(&keypad->mutex);
    spin_lock_init(&keypad->spinlock);

    /* Create and register the input driver. */
    input_dev = input_allocate_device();
    if (!input_dev) {
        dev_err(&pdev->dev, "failed to allocate input device\n");
        error = -ENOMEM;
        goto failed_free;
    }

    input_dev->name = pdev->name;
    input_dev->id.bustype = BUS_HOST;
    input_dev->open = android_keypad_open;
    input_dev->close = android_keypad_close;
    input_dev->dev.parent = &pdev->dev;

    keypad->input_dev = input_dev;
    input_set_drvdata(input_dev, keypad);

    set_bit(EV_KEY, input_dev->evbit);
    set_bit(EV_REL, input_dev->evbit);

    android_keypad_setkeycode(keypad);
    platform_set_drvdata(pdev, keypad);

    /* Register the input device */
    error = input_register_device(input_dev);
    if (error) {
        dev_err(&pdev->dev, "failed to register input device\n");
        goto failed_free_dev;
    }

    return 0;

```

```

failed_free_dev:
    platform_set_drvdata(pdev, NULL);
    input_free_device(input_dev);
failed_free:
    kfree(keypad);
    return error;
}

static int __devexit android_keypad_remove(struct platform_device *pdev)
{
    struct android_keypad *keypad = platform_get_drvdata(pdev);

    input_unregister_device(keypad->input_dev);
    input_free_device(keypad->input_dev);

    platform_set_drvdata(pdev, NULL);
    kfree(keypad);
    return 0;
}

static struct platform_driver android_keypad_driver = {
    .probe      = android_keypad_probe,
    .remove     = __devexit_p(android_keypad_remove),
    .suspend    = android_keypad_suspend,
    .resume     = android_keypad_resume,
    .driver     = {
        .name    = "android-keypad",
    },
};

static int __init android_keypad_init(void)
{
    return platform_driver_register(&android_keypad_driver);
}

static void __exit android_keypad_exit(void)
{

```

```
    platform_driver_unregister(&android_keypad_driver);  
}  
  
module_init(android_keypad_init);  
module_exit(android_keypad_exit);  
  
MODULE_DESCRIPTION("Android Keypad Controller Driver");  
MODULE_LICENSE("GPL");
```



附錄E Touch driver (ucb1400_ts.c)

```
diff -raNu kernel.git/drivers/input/touchscreen/ucb1400_ts.c
linux-2.6.25-android-1.0_r1/drivers/input/touchscreen/ucb1400_ts.c
--- kernel.git/drivers/input/touchscreen/ucb1400_ts.c    2008-07-24
09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/drivers/input/touchscreen/ucb1400_ts.c    2009-05-15
17:07: 29.000000000 +0800
@@ -1,6 +1,9 @@
/*
 * Philips UCB1400 touchscreen driver
 *
+ * Author: Mask <cycdisk@gmail.com>
+ * Modified: April, 2009
+ *
 * Author: Nicolas Pitre
 * Created: September 25, 2006
 * Copyright: MontaVista Software, Inc.
@@ -33,7 +36,6 @@
/*
 * Interesting UCB1400 AC-link registers
 */
-
#define UCB_IE_RIS                0x5e
#define UCB_IE_FAL                0x60
#define UCB_IE_STATUS            0x62
@@ -97,6 +99,10 @@
static int adcsync;
static int ts_delay = 55; /* us */
static int ts_delay_pressure; /* us */
+static u16 g_min_x = 90;
+static u16 g_min_y = 80;
+static u16 g_max_x = 920;
+static u16 g_max_y = 950;

static inline u16 ucb1400_reg_read(struct ucb1400 *ucb, u16 reg)
{
```

@@ -252,14 +258,28 @@

```
static void ucb1400_ts_evt_add(struct input_dev *idev, u16 pressure, u16 x, u16 y)
{
+/* it's hard code here, may be you should modify */
+    if(x > g_max_x)
+        x = g_max_x - 1;
+    else if(x < g_min_x)
+        x = g_min_x + 1;
+    if(y > g_max_y)
+        y = g_max_y - 1;
+    else if(y < g_min_y)
+        y = g_min_y + 1;
+    x = ((x - g_min_x) * 239) / (g_max_x - g_min_x);
+    y = ((y - g_min_y) * 319) / (g_max_y - g_min_y);
+    y = 320 - y;
+    input_report_key(idev, BTN_TOUCH, 1);
+    input_report_abs(idev, ABS_X, x);
+    input_report_abs(idev, ABS_Y, y);
-    input_report_abs(idev, ABS_PRESSURE, pressure);
+    input_report_abs(idev, ABS_PRESSURE, pressure ? 1 : 0);
+    input_sync(idev);
}
```

```
static void ucb1400_ts_event_release(struct input_dev *idev)
{
+    input_report_key(idev, BTN_TOUCH, 0);
+    input_report_abs(idev, ABS_PRESSURE, 0);
+    input_sync(idev);
}
```

@@ -310,7 +330,7 @@

```
    /* Switch back to interrupt mode. */
    ucb1400_ts_mode_int(ucb);

-    msleep(10);
+    msleep(1);

    if (ucb1400_ts_pen_down(ucb)) {
```

```

                                ucb1400_ts_irq_enable(ucb);
@@ -328,7 +348,7 @@
                                } else {
                                    valid = 1;
                                    ucb1400_ts_evt_add(ucb->ts_iddev, p, x, y);
-                                    timeout = msecs_to_jiffies(10);
+                                    timeout = msecs_to_jiffies(0);
                                }

                                wait_event_freezable_timeout(ucb->ts_wait,
@@ -427,10 +447,12 @@
                                unsigned long mask, timeout;

                                mask = probe_irq_on();
+                                /*
                                if (!mask) {
                                    probe_irq_off(mask);
                                    return -EBUSY;
                                }
+                                */

                                /* Enable the ADC interrupt. */
                                ucb1400_reg_write(ucb, UCB_IE_RIS, UCB_IE_ADC);
@@ -463,7 +485,10 @@
                                /* Read triggered interrupt. */
                                ucb->irq = probe_irq_off(mask);
                                if (ucb->irq < 0 || ucb->irq == NO_IRQ)
-                                    return -ENODEV;
+                                {
+                                    ucb->irq = CREATOR_TOUCH_IRQ;
+                                    //return -ENODEV;
+                                }

                                return 0;
                            }
@@ -498,7 +523,8 @@
                                goto err_free_devs;
                            }

```

```

-     error = request_irq(ucb->irq, ucb1400_hard_irq, IRQF_TRIGGER_RISING,
+     //error = request_irq(ucb->irq, ucb1400_hard_irq, IRQF_TRIGGER_RISING,
+     error = request_irq(ucb->irq, ucb1400_hard_irq, 0,
                        "UCB1400", ucb);

    if (error) {
        printk(KERN_ERR "ucb1400: unable to grab irq%d: %d\n",
@@ -515,21 +541,27 @@
        idev->id.product        = id;
        idev->open              = ucb1400_ts_open;
        idev->close             = ucb1400_ts_close;
-     idev->evbit[0]           = BIT_MASK(EV_ABS);
+     set_bit(EV_KEY,idev->evbit);
+     set_bit(EV_ABS,idev->evbit);
+     set_bit(BTN_TOUCH,idev->keybit);

    ucb1400_adc_enable(ucb);
    x_res = ucb1400_ts_read_xres(ucb);
    y_res = ucb1400_ts_read_yres(ucb);
    ucb1400_adc_disable(ucb);
-     printk(KERN_DEBUG "UCB1400: x/y = %d/%d\n", x_res, y_res);
+     printk("UCB1400: x/y = %d/%d\n", x_res, y_res);

-     input_set_abs_params(idev, ABS_X, 0, x_res, 0, 0);
-     input_set_abs_params(idev, ABS_Y, 0, y_res, 0, 0);
-     input_set_abs_params(idev, ABS_PRESSURE, 0, 0, 0, 0);
+/* it's hard code here, may be you should modify */
+     input_set_abs_params(idev, ABS_X, 0, 240, 0, 0);
+     input_set_abs_params(idev, ABS_Y, 0, 320, 0, 0);
+     input_set_abs_params(idev, ABS_PRESSURE, 0, 1, 0, 0);

    error = input_register_device(idev);
    if (error)
+     {
+         printk("%s:%s input_register_device FAIL\n",__FILE__,__func__);
+         goto err_free_irq;
+     }

```

```
dev_set_drvdata(dev, ucb);  
return 0;
```



附錄Finit.rc

```
--- ../../mydroid/cdma-import/out/target/product/generic/root/init.rc    2009-04-23
16:29:40.000000000 +0800
+++ init.rc    2009-05-15 17:11:11.000000000 +0800
@@ -14,17 +14,17 @@
    export BOOTCLASSPATH
/system/framework/core.jar:/system/framework/ext.jar:/system/framework/framework.j
ar:/system/framework/android.policy.jar:/system/framework/services.jar

# Backward compatibility
-    symlink /system/etc /etc
+#    symlink /system/etc /etc

# create mountpoints and mount tmpfs on sqlite_stmt_journals
-    mkdir /sdcard 0000 system system
-    mkdir /system
-    mkdir /data 0771 system system
-    mkdir /cache 0770 system cache
-    mkdir /sqlite_stmt_journals 01777 root root
-    mount tmpfs tmpfs /sqlite_stmt_journals size=4m
-
-    mount rootfs rootfs / ro remount
+#    mkdir /sdcard 0000 system system
+#    mkdir /system
+#    mkdir /data 0771 system system
+#    mkdir /cache 0770 system cache
+#    mkdir /sqlite_stmt_journals 01777 root root
+#    mount tmpfs tmpfs /sqlite_stmt_journals size=4m
+#
+#    mount rootfs rootfs / ro remount

write /proc/sys/kernel/panic_on_oops 1
write /proc/sys/kernel/hung_task_timeout_secs 0
@@ -34,22 +34,22 @@

# mount mtd partitions
```

```

# Mount /system rw first to give the filesystem a chance to save a checkpoint
- mount yaffs2 mtd@system /system
- mount yaffs2 mtd@system /system ro remount
-
- # We chown/chmod /data again so because mount is run as root + defaults
- mount yaffs2 mtd@userdata /data nosuid nodev
- chown system system /data
- chmod 0771 /data
-
- # Same reason as /data above
- mount yaffs2 mtd@cache /cache nosuid nodev
- chown system cache /cache
- chmod 0770 /cache
-
- # This may have been created by the recovery system with odd permissions
- chown system system /cache/recovery
- chmod 0770 /cache/recovery
+# mount yaffs2 mtd@system /system
+# mount yaffs2 mtd@system /system ro remount
+#
+# # We chown/chmod /data again so because mount is run as root + defaults
+# mount yaffs2 mtd@userdata /data nosuid nodev
+# chown system system /data
+# chmod 0771 /data
+#
+# # Same reason as /data above
+# mount yaffs2 mtd@cache /cache nosuid nodev
+# chown system cache /cache
+# chmod 0770 /cache
+#
+# # This may have been created by the recovery system with odd permissions
+# chown system system /cache/recovery
+# chmod 0770 /cache/recovery

# create basic filesystem structure
mkdir /data/misc 01771 system misc
@@ -115,34 +115,34 @@
# Permissions for System Server and daemons.

```

```

chown radio system /sys/android_power/state
chown radio system /sys/android_power/request_state
- chown radio system /sys/android_power/acquire_full_wake_lock
- chown radio system /sys/android_power/acquire_partial_wake_lock
- chown radio system /sys/android_power/release_wake_lock
- chown system system /sys/class/timed_output/vibrator/enable
- chown system system /sys/class/leds/keyboard-backlight/brightness
- chown system system /sys/class/leds/lcd-backlight/brightness
- chown system system /sys/class/leds/button-backlight/brightness
- chown system system /sys/class/leds/red/brightness
- chown system system /sys/class/leds/green/brightness
- chown system system /sys/class/leds/blue/brightness
- chown system system /sys/class/leds/red/device/grpfreq
- chown system system /sys/class/leds/red/device/grppwm
- chown system system /sys/class/leds/red/device/blink
- chown system system /sys/class/leds/red/brightness
- chown system system /sys/class/leds/green/brightness
- chown system system /sys/class/leds/blue/brightness
- chown system system /sys/class/leds/red/device/grpfreq
- chown system system /sys/class/leds/red/device/grppwm
- chown system system /sys/class/leds/red/device/blink
- chown system system /sys/class/timed_output/vibrator/enable
- chown system system /sys/module/sco/parameters/disable_esco
- chown system system /sys/kernel/ipv4/tcp_wmem_min
- chown system system /sys/kernel/ipv4/tcp_wmem_def
- chown system system /sys/kernel/ipv4/tcp_wmem_max
- chown system system /sys/kernel/ipv4/tcp_rmem_min
- chown system system /sys/kernel/ipv4/tcp_rmem_def
- chown system system /sys/kernel/ipv4/tcp_rmem_max
- chown root radio /proc/cmdline
+# chown radio system /sys/android_power/acquire_full_wake_lock
+# chown radio system /sys/android_power/acquire_partial_wake_lock
+# chown radio system /sys/android_power/release_wake_lock
+# chown system system /sys/class/timed_output/vibrator/enable
+# chown system system /sys/class/leds/keyboard-backlight/brightness
+# chown system system /sys/class/leds/lcd-backlight/brightness
+# chown system system /sys/class/leds/button-backlight/brightness
+# chown system system /sys/class/leds/red/brightness

```

```

+# chown system system /sys/class/leds/green/brightness
+# chown system system /sys/class/leds/blue/brightness
+# chown system system /sys/class/leds/red/device/grpfreq
+# chown system system /sys/class/leds/red/device/grppwm
+# chown system system /sys/class/leds/red/device/blink
+# chown system system /sys/class/leds/red/brightness
+# chown system system /sys/class/leds/green/brightness
+# chown system system /sys/class/leds/blue/brightness
+# chown system system /sys/class/leds/red/device/grpfreq
+# chown system system /sys/class/leds/red/device/grppwm
+# chown system system /sys/class/leds/red/device/blink
+# chown system system /sys/class/timed_output/vibrator/enable
+# chown system system /sys/module/sco/parameters/disable_esco
+# chown system system /sys/kernel/ipv4/tcp_wmem_min
+# chown system system /sys/kernel/ipv4/tcp_wmem_def
+# chown system system /sys/kernel/ipv4/tcp_wmem_max
+# chown system system /sys/kernel/ipv4/tcp_rmem_min
+# chown system system /sys/kernel/ipv4/tcp_rmem_def
+# chown system system /sys/kernel/ipv4/tcp_rmem_max
+# chown root radio /proc/cmdline

```

Define TCP buffer sizes for various networks

```

# ReadMin, ReadInitial, ReadMax, WriteMin, WriteInitial, WriteMax,
@@ -184,15 +184,15 @@

```

service debuggerd /system/bin/debuggerd

-service ril-daemon /system/bin/rild

```

- socket rild stream 660 root radio
- socket rild-debug stream 660 radio system
- user root
- group radio cache inet misc

```

+#service ril-daemon /system/bin/rild

```

+# socket rild stream 660 root radio
+# socket rild-debug stream 660 radio system
+# user root
+# group radio cache inet misc

```

```

service zygote /system/bin/app_process -Xzygote /system/bin --zygote
--start-system-server
    socket zygote stream 666
-    onrestart write /sys/android_power/request_state wake
+#    onrestart write /sys/android_power/request_state wake

service media /system/bin/mediaserver
    user media
@@ -203,30 +203,30 @@
    group audio
    oneshot

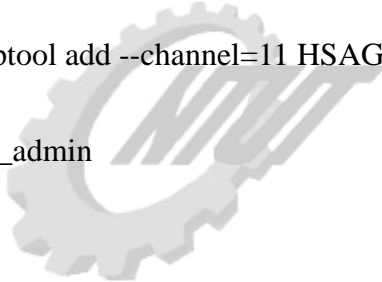
-service dbus /system/bin/dbus-daemon --system --nofork
-    socket dbus stream 660 bluetooth bluetooth
-    user bluetooth
-    group bluetooth net_bt_admin
-
-service hcid /system/bin/hcid -s -n -f /etc/hcid.conf
-    socket bluetooth stream 660 bluetooth bluetooth
-    socket dbus_bluetooth stream 660 bluetooth bluetooth
-    # init.rc does not yet support applying capabilities, so run as root and
-    # let hcid drop uid to bluetooth with the right linux capabilities
-    group bluetooth net_bt_admin misc
-    disabled
-
-service hfag /system/bin/sdptool add --channel=10 HFAG
-    user bluetooth
-    group bluetooth net_bt_admin
-    disabled
-    oneshot
-
-service hsag /system/bin/sdptool add --channel=11 HSAG
-    user bluetooth
-    group bluetooth net_bt_admin
-    disabled
-    oneshot
+#service dbus /system/bin/dbus-daemon --system --nofork
+#    socket dbus stream 660 bluetooth bluetooth

```

```

+#    user bluetooth
+#    group bluetooth net_bt_admin
+#
+#service hcid /system/bin/hcid -s -n -f /etc/hcid.conf
+#    socket bluetooth stream 660 bluetooth bluetooth
+#    socket dbus_bluetooth stream 660 bluetooth bluetooth
+#    # init.rc does not yet support applying capabilities, so run as root and
+#    # let hcid drop uid to bluetooth with the right linux capabilities
+#    group bluetooth net_bt_admin misc
+#    disabled
+#
+#service hfag /system/bin/sdptool add --channel=10 HFAG
+#    user bluetooth
+#    group bluetooth net_bt_admin
+#    disabled
+#    oneshot
+#
+#service hsag /system/bin/sdptool add --channel=11 HSAG
+#    user bluetooth
+#    group bluetooth net_bt_admin
+#    disabled
+#    oneshot

```



```

service installd /system/bin/installd
    socket installd stream 600 system system

```

附錄G linux-2.6.25-android-1.0_r1/Makefile

```
diff -raNu kernel.git/Makefile linux-2.6.25-android-1.0_r1/Makefile
--- kernel.git/Makefile 2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/Makefile      2009-05-15 14:13:22.000000000
+0800
@@ -1,7 +1,7 @@
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 25
-EXTRAVERSION =
+EXTRAVERSION = [cycdisk@gmail.com]
NAME = Funky Weasel is Jiggy wit it

# *DOCUMENTATION*
@@ -192,7 +192,7 @@
# Note: Some architectures assign CROSS_COMPILE in their arch/*/Makefile
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH          ?= $(SUBARCH)
-CROSS_COMPILE ?= arm-eabi-
+CROSS_COMPILE ?=

# Architecture as present in compile.h
UTS_MACHINE   := $(ARCH)
```

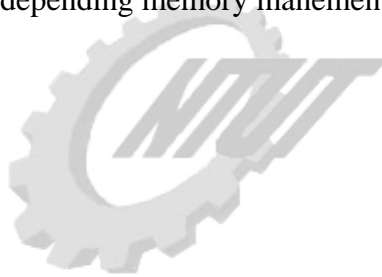
附錄H linux-2.6.25-android-1.0_r1/arch/arm/Makefile

e

```
diff -raNu kernel.git/arch/arm/Makefile linux-2.6.25-android-1.0_r1/arch/arm/Makefile
--- kernel.git/arch/arm/Makefile      2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/arch/arm/Makefile      2009-01-20
00:53:48.000000000 +0800
@@ -20,7 +20,7 @@
```

```
# Do not use arch/arm/defconfig - it's always outdated.
# Select a platform tht is kept up-to-date
-KBUILD_DEFCONFIG := versatile_defconfig
+KBUILD_DEFCONFIG := android_pxa270_defconfig

# defines filename extension depending memory manement type.
ifeq ($(CONFIG_MMU),)
```



附錄I **linux-2.6.25-android-1.0_r1/arch/arm/configs** **/android_pxa270_defconfig**

因為礙於文章篇幅，所以我們僅列出第一版與最後一版的差異。

Index: arch/arm/configs/android_pxa270_defconfig

```
=====
=====
--- arch/arm/configs/android_pxa270_defconfig    (.../linux-2.6.25-android-1.0_r1/
arch/arm/configs/android_pxa270_defconfig)    (revision 1)
+++ arch/arm/configs/android_pxa270_defconfig
(.../kernel/linux-2.6.25-android-1.0_r1/arch/arm/configs/android_pxa270_defconfig)
(revision 147)
@@ -1,7 +1,7 @@
#
# Automatically generated make config: don't edit
-# Linux kernel version: 2.6.25
-# Tue Dec 23 16:30:14 2008
+# Linux kernel version: 2.6.25[cycdisk@gmail.com]
+# Sat May 16 23:38:41 2009
#
CONFIG_ARM=y
CONFIG_SYS_SUPPORTS_APM_EMULATION=y
@@ -35,7 +35,7 @@
CONFIG_BROKEN_ON_SMP=y
CONFIG_INIT_ENV_ARG_LIMIT=32
CONFIG_LOCALVERSION=""
-CONFIG_LOCALVERSION_AUTO=y
+# CONFIG_LOCALVERSION_AUTO is not set
CONFIG_SWAP=y
CONFIG_SYSVIPC=y
CONFIG_SYSVIPC_SYSCTL=y
@@ -44,7 +44,7 @@
# CONFIG_TASKSTATS is not set
# CONFIG_AUDIT is not set
# CONFIG_IKCONFIG is not set
```

```

-CONFIG_LOG_BUF_SHIFT=14
+CONFIG_LOG_BUF_SHIFT=16
# CONFIG_CGROUPS is not set
CONFIG_GROUP_SCHED=y
CONFIG_FAIR_GROUP_SCHED=y
@@ -82,7 +82,7 @@
CONFIG_TIMERFD=y
CONFIG_EVENTFD=y
CONFIG_SHMEM=y
-# CONFIG_ASHMEM is not set
+CONFIG_ASHMEM=y
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLUB_DEBUG=y
# CONFIG_SLAB is not set
@@ -220,18 +220,9 @@
#
# CONFIG_PCI_SYSCALL is not set
# CONFIG_ARCH_SUPPORTS_MSI is not set
-CONFIG_PCCARD=y
-# CONFIG_PCMCIA_DEBUG is not set
-CONFIG_PCMCIA=y
-CONFIG_PCMCIA_LOAD_CIS=y
-CONFIG_PCMCIA_IOCTL=y
+# CONFIG_PCCARD is not set

#
-# PC-card bridges
-#
-CONFIG_PCMCIA_PXA2XX=y
-
-#
# Kernel Features
#
CONFIG_TICK_ONESHOT=y
@@ -290,7 +281,13 @@
#
# Power management options
#

```

```
-# CONFIG_PM is not set
+CONFIG_PM=y
+# CONFIG_PM_LEGACY is not set
+# CONFIG_PM_DEBUG is not set
+CONFIG_PM_SLEEP=y
+CONFIG_SUSPEND=y
+CONFIG_SUSPEND_FREEZER=y
+# CONFIG_APM_EMULATION is not set
CONFIG_ARCH_SUSPEND_POSSIBLE=y
```

```
#
@@ -393,10 +390,6 @@
# CONFIG_BT_HCIBCM203X is not set
# CONFIG_BT_HCIBPA10X is not set
# CONFIG_BT_HCIBFUSB is not set
-# CONFIG_BT_HCIDTL1 is not set
-# CONFIG_BT_HCIBT3C is not set
-# CONFIG_BT_HCIBLUECARD is not set
-# CONFIG_BT_HCIBTUART is not set
# CONFIG_BT_HCIVHCI is not set
# CONFIG_AF_RXRPC is not set
```


```
@@ -426,7 +419,7 @@
CONFIG_STANDALONE=y
CONFIG_PREVENT_FIRMWARE_BUILD=y
CONFIG_FW_LOADER=y
-CONFIG_DEBUG_DRIVER=y
+# CONFIG_DEBUG_DRIVER is not set
# CONFIG_DEBUG_DEVRES is not set
# CONFIG_SYS_HYPERVISOR is not set
# CONFIG_CONNECTOR is not set
```

```
@@ -539,7 +532,6 @@
# CONFIG_BLK_DEV_IDE_SATA is not set
CONFIG_BLK_DEV_IDEDISK=y
CONFIG_IDEDISK_MULTI_MODE=y
-CONFIG_BLK_DEV_IDECS=y
# CONFIG_BLK_DEV_IDECD is not set
# CONFIG_BLK_DEV_IDETAPE is not set
```

```

# CONFIG_BLK_DEV_IDEFLOPPY is not set
@@ -597,7 +589,6 @@
CONFIG_SCSI_LOWLEVEL=y
# CONFIG_ISCSI_TCP is not set
# CONFIG_SCSI_DEBUG is not set
-# CONFIG_SCSI_LOWLEVEL_PCMCIA is not set
# CONFIG_ATA is not set
# CONFIG_MD is not set
CONFIG_NETDEVICES=y
@@ -638,7 +629,6 @@
# CONFIG_USB_PEGASUS is not set
# CONFIG_USB_RTL8150 is not set
# CONFIG_USB_USBNET is not set
-# CONFIG_NET_PCMCIA is not set
# CONFIG_WAN is not set
# CONFIG_PPP is not set
# CONFIG_SLIP is not set
@@ -668,11 +658,31 @@
#
# Input device drivers
#
-# CONFIG_INPUT_KEYBOARD is not set
+CONFIG_INPUT_KEYBOARD=y
+# CONFIG_KEYBOARD_ATKBD is not set
+# CONFIG_KEYBOARD_SUNKBD is not set
+# CONFIG_KEYBOARD_LKKBD is not set
+# CONFIG_KEYBOARD_XTKBD is not set
+# CONFIG_KEYBOARD_NEWTON is not set
+# CONFIG_KEYBOARD_STOWAWAY is not set
+# CONFIG_KEYBOARD_PXA27x is not set
+CONFIG_KEYBOARD_ANDROID=y
+# CONFIG_KEYBOARD_GPIO is not set
+# CONFIG_KEYBOARD_GOLDFISH_EVENTS is not set
# CONFIG_INPUT_MOUSE is not set
# CONFIG_INPUT_JOYSTICK is not set
# CONFIG_INPUT_TABLET is not set
-# CONFIG_INPUT_TOUCHSCREEN is not set
+CONFIG_INPUT_TOUCHSCREEN=y

```



```

+# CONFIG_TOUCHSCREEN_FUJITSU is not set
+# CONFIG_TOUCHSCREEN_GUNZE is not set
+# CONFIG_TOUCHSCREEN_ELO is not set
+# CONFIG_TOUCHSCREEN_MTOUCH is not set
+# CONFIG_TOUCHSCREEN_MK712 is not set
+# CONFIG_TOUCHSCREEN_PENMOUNT is not set
+# CONFIG_TOUCHSCREEN_TOUCHRIGHT is not set
+# CONFIG_TOUCHSCREEN_TOUCHWIN is not set
+CONFIG_TOUCHSCREEN_UCB1400=y
+# CONFIG_TOUCHSCREEN_USB_COMPOSITE is not set
# CONFIG_INPUT_MISC is not set

#
@@ -709,14 +719,6 @@
CONFIG_HW_RANDOM=m
# CONFIG_NVRAM is not set
# CONFIG_R3964 is not set
-
-#
-# PCMCIA character devices
-#
-# CONFIG_SYNCLINK_CS is not set
-# CONFIG_CARDMAN_4000 is not set
-# CONFIG_CARDMAN_4040 is not set
-# CONFIG_IPWIRELESS is not set
# CONFIG_RAW_DRIVER is not set
# CONFIG_TCG_TPM is not set
# CONFIG_CREATOR_PXA270_LCD is not set
@@ -761,6 +763,7 @@
# CONFIG_SENSORS_PCF8574 is not set
# CONFIG_PCF8575 is not set
# CONFIG_SENSORS_PCF8591 is not set
+# CONFIG_SENSORS_RTC8564 is not set
# CONFIG_TPS65010 is not set
# CONFIG_SENSORS_MAX6875 is not set
# CONFIG_SENSORS_TSL2550 is not set
@@ -907,9 +910,9 @@
CONFIG_FB=y

```

```

# CONFIG_FIRMWARE_EDID is not set
# CONFIG_FB_DDC is not set
-CONFIG_FB_CFB_FILLRECT=m
-CONFIG_FB_CFB_COPYAREA=m
-CONFIG_FB_CFB_IMAGEBLIT=m
+CONFIG_FB_CFB_FILLRECT=y
+CONFIG_FB_CFB_COPYAREA=y
+CONFIG_FB_CFB_IMAGEBLIT=y
# CONFIG_FB_CFB_REV_PIXELS_IN_BYTE is not set
# CONFIG_FB_SYS_FILLRECT is not set
# CONFIG_FB_SYS_COPYAREA is not set
@@ -926,7 +929,11 @@
# Frame buffer hardware drivers
#
# CONFIG_FB_S1D13XXX is not set
-CONFIG_FB_PXA=m
+CONFIG_FB_PXA=y
+# CONFIG_MTLCD_0283224 is not set
+CONFIG_MTLCD_0353224=y
+# CONFIG_MTLCD_0353224A is not set
+# CONFIG_MTLCD_1046448 is not set
# CONFIG_FB_PXA_PARAMETERS is not set
# CONFIG_FB_MBX is not set
# CONFIG_FB_GOLDFISH is not set
@@ -1004,12 +1011,6 @@
# CONFIG_SND_USB_CAIAQ is not set

#
-# PCMCIA devices
-#
-# CONFIG_SND_VXPOCKET is not set
-# CONFIG_SND_PDAUDIOCF is not set
-
-#
# System on Chip audio support
#
# CONFIG_SND_SOC is not set
@@ -1053,6 +1054,8 @@

```

```

# CONFIG_USB_DEVICEFS is not set
CONFIG_USB_DEVICE_CLASS=y
# CONFIG_USB_DYNAMIC_MINORS is not set
+# CONFIG_USB_SUSPEND is not set
+# CONFIG_USB_PERSIST is not set
# CONFIG_USB_OTG is not set

#
@@ -1090,7 +1093,6 @@
# CONFIG_USB_STORAGE_SDDR55 is not set
# CONFIG_USB_STORAGE_JUMPSHOT is not set
# CONFIG_USB_STORAGE_ALAUDA is not set
-# CONFIG_USB_STORAGE_ONETOUCH is not set
# CONFIG_USB_STORAGE_KARMA is not set
# CONFIG_USB_LIBUSUAL is not set

@@ -1146,12 +1148,62 @@
# CONFIG_NEW_LEDS is not set
# CONFIG_SWITCH is not set
CONFIG_RTC_LIB=y
-# CONFIG_RTC_CLASS is not set
+CONFIG_RTC_CLASS=y
+# CONFIG_RTC_HCTOSYS is not set
+# CONFIG_RTC_DEBUG is not set

#
+# RTC interfaces
+#
+# CONFIG_RTC_INTF_SYSFS is not set
+# CONFIG_RTC_INTF_PROC is not set
+# CONFIG_RTC_INTF_DEV is not set
+# CONFIG_RTC_DRV_TEST is not set
+
+
+#
+# I2C RTC drivers
+#
+# CONFIG_RTC_DRV_DS1307 is not set
+# CONFIG_RTC_DRV_DS1374 is not set

```

```

+# CONFIG_RTC_DRV_DS1672 is not set
+# CONFIG_RTC_DRV_MAX6900 is not set
+# CONFIG_RTC_DRV_RS5C372 is not set
+# CONFIG_RTC_DRV_ISL1208 is not set
+# CONFIG_RTC_DRV_X1205 is not set
+# CONFIG_RTC_DRV_PCF8563 is not set
+# CONFIG_RTC_DRV_PCF8583 is not set
+# CONFIG_RTC_DRV_M41T80 is not set
+# CONFIG_RTC_DRV_S35390A is not set
+
+#
+# SPI RTC drivers
+#
+
+#
+# Platform RTC drivers
+#
+# CONFIG_RTC_DRV_CMOS is not set
+# CONFIG_RTC_DRV_DS1511 is not set
+# CONFIG_RTC_DRV_DS1553 is not set
+# CONFIG_RTC_DRV_DS1742 is not set
+# CONFIG_RTC_DRV_STK17TA8 is not set
+# CONFIG_RTC_DRV_M48T86 is not set
+# CONFIG_RTC_DRV_M48T59 is not set
+# CONFIG_RTC_DRV_V3020 is not set
+
+#
+# on-CPU RTC drivers
+#
+# CONFIG_RTC_DRV_SA1100 is not set
+# CONFIG_RTC_DRV_GOLDFISH is not set
+
+#
# Android
#
# CONFIG_ANDROID_RAM_CONSOLE is not set
+CONFIG_ANDROID_POWER=y
+CONFIG_ANDROID_POWER_ALARM=y

```



```

+CONFIG_ANDROID_POWER_STAT=y
CONFIG_ANDROID_LOGGER=y
CONFIG_ANDROID_TIMED_GPIO=y
CONFIG_ANDROID_BINDER_IPC=y
@@ -1163,8 +1215,13 @@
CONFIG_EXT2_FS=y
# CONFIG_EXT2_FS_XATTR is not set
# CONFIG_EXT2_FS_XIP is not set
-# CONFIG_EXT3_FS is not set
+CONFIG_EXT3_FS=y
+CONFIG_EXT3_FS_XATTR=y
+# CONFIG_EXT3_FS_POSIX_ACL is not set
+# CONFIG_EXT3_FS_SECURITY is not set
# CONFIG_EXT4DEV_FS is not set
+CONFIG_JBD=y
+CONFIG_FS_MBCACHE=y
# CONFIG_REISERFS_FS is not set
# CONFIG_JFS_FS is not set
# CONFIG_FS_POSIX_ACL is not set
@@ -1216,7 +1273,16 @@
# CONFIG_BEFS_FS is not set
# CONFIG_BFS_FS is not set
# CONFIG_EFS_FS is not set
-# CONFIG_YAFFS_FS is not set
+CONFIG_YAFFS_FS=y
+CONFIG_YAFFS_YAFFS1=y
+# CONFIG_YAFFS_9BYTE_TAGS is not set
+# CONFIG_YAFFS_DOES_ECC is not set
+CONFIG_YAFFS_YAFFS2=y
+CONFIG_YAFFS_AUTO_YAFFS2=y
+# CONFIG_YAFFS_DISABLE_LAZY_LOAD is not set
+# CONFIG_YAFFS_DISABLE_WIDE_TNODES is not set
+# CONFIG_YAFFS_ALWAYS_CHECK_CHUNK_ERASED is not set
+CONFIG_YAFFS_SHORT_NAMES_IN_RAM=y
CONFIG_JFFS2_FS=y
CONFIG_JFFS2_FS_DEBUG=0
CONFIG_JFFS2_FS_WRITEBUFFER=y
@@ -1349,9 +1415,9 @@


```

```
# CONFIG_LOCK_STAT is not set
# CONFIG_DEBUG_SPINLOCK_SLEEP is not set
# CONFIG_DEBUG_LOCKING_API_SELFTESTS is not set
-CONFIG_DEBUG_KOBJECT=y
+# CONFIG_DEBUG_KOBJECT is not set
  CONFIG_DEBUG_BUGVERBOSE=y
-CONFIG_DEBUG_INFO=y
+# CONFIG_DEBUG_INFO is not set
  # CONFIG_DEBUG_VM is not set
  # CONFIG_DEBUG_LIST is not set
  # CONFIG_DEBUG_SG is not set
@@ -1363,10 +1429,9 @@
  # CONFIG_LATENCYTOP is not set
  # CONFIG_SAMPLES is not set
  CONFIG_DEBUG_USER=y
-CONFIG_DEBUG_ERRORS=y
+# CONFIG_DEBUG_ERRORS is not set
  # CONFIG_DEBUG_STACK_USAGE is not set
-CONFIG_DEBUG_LL=y
-# CONFIG_DEBUG_ICEDCC is not set
+# CONFIG_DEBUG_LL is not set

#
# Security options
```

附錄J **linux-2.6.25-android-1.0_r1/arch/arm/kernel/** **head.S**

```
diff -raNu kernel.git/arch/arm/kernel/head.S
linux-2.6.25-android-1.0_r1/arch/arm/kernel/head.S
--- kernel.git/arch/arm/kernel/head.S    2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/arch/arm/kernel/head.S    2009-01-13
22:18:17.000000000 +0800
@@ -77,6 +77,7 @@
        .section ".text.head", "ax"
        .type     stext, %function
ENTRY(stext)
+       LDR       r1,=0x2d9
        msr       cpsr_c, #PSR_F_BIT | PSR_I_BIT | SVC_MODE @ ensure svc
mode
        mrc       p15, 0, r9, c0, c0 @ and irq disabled
                                @ get processor id
```



附錄K linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/clock.c

```
diff -raNu kernel.git/arch/arm/mach-pxa/clock.c
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/clock.c
--- kernel.git/arch/arm/mach-pxa/clock.c      2008-07-24 09:04:04.000000000
+0800
+++ linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/clock.c      2009-01-13
22:18:17.      0000000000 +0800
@@ -99,7 +99,7 @@
```

```
static void clk_gpio27_enable(struct clk *clk)
{
-    pxa_gpio_mode(GPIO11_3_6MHz_MD);
+    //pxa_gpio_mode(GPIO11_3_6MHz_MD);
}

static void clk_gpio27_disable(struct clk *clk)
```

附錄L **linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/irq.c**

```
diff -raNu kernel.git/arch/arm/mach-pxa/irq.c
linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/irq.c
--- kernel.git/arch/arm/mach-pxa/irq.c   2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/arch/arm/mach-pxa/irq.c 2009-03-03
00:53:47.000000000 +0800
@@ -96,7 +96,8 @@
        ICMR2 = 0;
        ICLR2 = 0;

-       for (irq = PXA_IRQ(32); irq < PXA_IRQ(64); irq++) {
+       for (irq = PXA_IRQ(32); irq < PXA_IRQ(34); irq++) {
+       //for (irq = PXA_IRQ(32); irq < PXA_IRQ(64); irq++) {
                set_irq_chip(irq, &pxa_internal_chip_high);
                set_irq_handler(irq, handle_level_irq);
                set_irq_flags(irq, IRQF_VALID);
```

附錄M linux-2.6.25-android-1.0_r1/drivers/cpufreq/

Kconfig

```
diff -raNu kernel.git/drivers/cpufreq/Kconfig
linux-2.6.25-android-1.0_r1/drivers/cpufreq/Kconfig
--- kernel.git/drivers/cpufreq/Kconfig 2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/drivers/cpufreq/Kconfig 2009-01-13
22:18:17.000000000 +0800
@@ -55,7 +55,7 @@

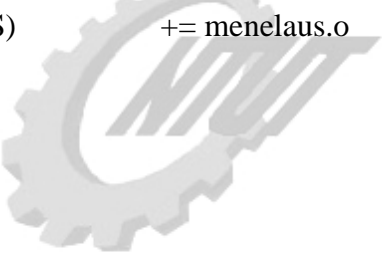
choice

    prompt "Default CPUFreq governor"
-    default CPU_FREQ_DEFAULT_GOV_USERSPACE if CPU_FREQ_SA1100
|| CPU_FREQ_SA1110
+    default CPU_FREQ_DEFAULT_GOV_USERSPACE if
CPU_FREQ_SA1100 || CPU_FREQ_SA1110 || CPU_FREQ_PXA27x
    default CPU_FREQ_DEFAULT_GOV_PERFORMANCE
    help
        This option sets which CPUFreq governor shall be loaded at
```

附錄N linux-2.6.25-android-1.0_r1/drivers/i2c/chips/

Makefile

```
diff -raNu kernel.git/drivers/i2c/chips/Makefile
linux-2.6.25-android-1.0_r1/drivers/i2c/chips/Makefile
--- kernel.git/drivers/i2c/chips/Makefile      2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/drivers/i2c/chips/Makefile      2009-03-03
00:53:47.      000000000 +0800
@@ -16,6 +16,7 @@
obj-$(CONFIG_SENSORS_PCF8574) += pcf8574.o
obj-$(CONFIG_PCF8575) += pcf8575.o
obj-$(CONFIG_SENSORS_PCF8591) += pcf8591.o
+obj-$(CONFIG_SENSORS_RTC8564) += rtc8564.o
obj-$(CONFIG_ISP1301_OMAP) += isp1301_omap.o
obj-$(CONFIG_TPS65010) += tps65010.o
obj-$(CONFIG_MENELAUS) += menelaus.o
```




附錄O linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Kconfig

```
diff -raNu kernel.git/drivers/input/keyboard/Kconfig
linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Kconfig
--- kernel.git/drivers/input/keyboard/Kconfig    2008-07-24 09:04:04.000000000
+0800
+++ linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Kconfig    2009-04-14
15:55:55.    000000000 +0800
@@ -268,6 +268,15 @@
    To compile this driver as a module, choose M here: the
    module will be called pxa27x_keypad.
```

```
+config KEYBOARD_ANDROID
+    tristate "Android keypad support"
+    depends on PXA27x
+    help
+        Enable support for Android keypad controller
+
+        To compile this driver as a module, choose M here: the
+        module will be called android_keypad.
+
+config KEYBOARD_AAED2000
+    tristate "AAED-2000 keyboard"
+    depends on MACH_AAED2000
```


附錄Plinux-2.6.25-android-1.0_r1/drivers/input/keyboard/Makefile

```
diff -raNu kernel.git/drivers/input/keyboard/Makefile
linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Makefile
--- kernel.git/drivers/input/keyboard/Makefile 2008-07-24 09:04:04.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/drivers/input/keyboard/Makefile 2009-04-14 15:55:55.
000000000 +0800
@@ -20,6 +20,7 @@
obj-$(CONFIG_KEYBOARD_HIL_OLD) += hilkbd.o
obj-$(CONFIG_KEYBOARD_OMAP) += omap-keypad.o
obj-$(CONFIG_KEYBOARD_PXA27x) += pxa27x_keypad.o
+obj-$(CONFIG_KEYBOARD_ANDROID) += android_keypad.o
obj-$(CONFIG_KEYBOARD_AAED2000) += aaed2000_kbd.o
obj-$(CONFIG_KEYBOARD_GPIO) += gpio_keys.o
obj-$(CONFIG_KEYBOARD_GOLDFISH_EVENTS) += goldfish_events.o
```



附錄Q linux-2.6.25-android-1.0_r1/include/linux/con fig.h

```
diff -raNu kernel.git/include/linux/config.h
linux-2.6.25-android-1.0_r1/include/linux/config.h
--- kernel.git/include/linux/config.h    1970-01-01 08:00:00.000000000 +0800
+++ linux-2.6.25-android-1.0_r1/include/linux/config.h  2009-01-13
22:18:17.000000000 +0800
@@ -0,0 +1,8 @@
+#ifndef _LINUX_CONFIG_H
+#define _LINUX_CONFIG_H
+/* This file is no longer in use and kept only for backward compatibility.
+ * autoconf.h is now included via -imacros on the commandline
+ */
+#include <linux/autoconf.h>
+
+#endif
```



附錄R mydroid/cdma-import/build/core/definitions. mk

Index: mydroid/cdma-import/build/core/definitions.mk

```
=====
=====
--- mydroid/cdma-import/build/core/definitions.mk      (revision 49)
+++ mydroid/cdma-import/build/core/definitions.mk      (revision 147)
@@ -549,7 +549,7 @@
define pretty
@echo $1
endef
-hide := @
+hide :=
else
define pretty
endef
```



附錄S **mydroid/cdma-import/external/sqlite/dist/Android.mk**

Index: mydroid/cdma-import/external/sqlite/dist/Android.mk

=====

--- mydroid/cdma-import/external/sqlite/dist/Android.mk (revision 49)
+++ mydroid/cdma-import/external/sqlite/dist/Android.mk (revision 147)
@@ -95,6 +95,7 @@

```
have_readline := $(wildcard /usr/include/readline/readline.h)
have_history := $(wildcard /usr/lib/libhistory*)
+have_termcap := $(wildcard /usr/lib/libtermcap*)
ifneq ($(strip $(have_readline)),)
LOCAL_CFLAGS += -DHAVE_READLINE=1
endif
@@ -107,6 +108,9 @@
ifneq ($(strip $(have_history)),)
LOCAL_LDLIBS += -lhistory
endif
+ifneq ($(strip $(have_termcap)),)
+LOCAL_LDLIBS += -ltermcap
+endif
```

```
LOCAL_MODULE := sqlite3
```

附錄T **mydroid/cdma-import/frameworks/base/core/ jni/server/com_android_server_BatteryService. e.cpp**

Index:

mydroid/cdma-import/frameworks/base/core/jni/server/com_android_server_BatteryService.
vice.cpp

```
=====
=====
---
mydroid/cdma-import/frameworks/base/core/jni/server/com_android_server_BatteryService.  
vice.cpp            (revision 49)
+++
mydroid/cdma-import/frameworks/base/core/jni/server/com_android_server_BatteryService.  
vice.cpp            (revision 147)
@@ -173,6 +173,7 @@

static void android_server_BatteryService_update(JNIEnv* env, jobject obj)
{
+/*
    setBooleanField(env, obj, AC_ONLINE_PATH, gFieldIds.mAcOnline);
    setBooleanField(env, obj, USB_ONLINE_PATH, gFieldIds.mUsbOnline);
    setBooleanField(env, obj, BATTERY_PRESENT_PATH,
gFieldIds.mBatteryPresent);
@@ -180,7 +181,15 @@
    setIntField(env, obj, BATTERY_CAPACITY_PATH, gFieldIds.mBatteryLevel);
    setIntField(env, obj, BATTERY_VOLTAGE_PATH, gFieldIds.mBatteryVoltage);
    setIntField(env, obj, BATTERY_TEMPERATURE_PATH,
gFieldIds.mBatteryTemperature);
-
+    */
+    env->SetBooleanField(obj, gFieldIds.mAcOnline,true);
+    env->SetBooleanField(obj, gFieldIds.mUsbOnline,true);
+    env->SetBooleanField(obj, gFieldIds.mBatteryPresent,true);
+    env->SetIntField(obj, gFieldIds.mBatteryLevel,100);
```

```

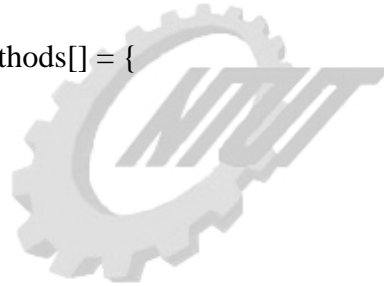
+         env->SetIntField(obj, gFieldIds.mBatteryVoltage,3);
+         env->SetIntField(obj, gFieldIds.mBatteryTemperature,10);
+
+         /*
+         const int SIZE = 128;
+         char buf[SIZE];

@@ -192,6 +201,9 @@

        if (readFromFile(BATTERY_TECHNOLOGY_PATH, buf, SIZE) > 0)
            env->SetObjectField(obj, gFieldIds.mBatteryTechnology,
env->NewStringUTF(buf));
+         */
+         env->SetIntField(obj, gFieldIds.mBatteryStatus,gConstants.statusCharging);
+         env->SetIntField(obj, gFieldIds.mBatteryHealth,gConstants.healthGood);
+     }

static JNINativeMethod sMethods[] = {

```



附錄U **mydroid/cdma-import/hardware/libhardware** **/power/power.c**

Index: mydroid/cdma-import/hardware/libhardware/power/power.c

```
=====
=====
--- mydroid/cdma-import/hardware/libhardware/power/power.c      (revision 49)
+++ mydroid/cdma-import/hardware/libhardware/power/power.c      (revision 147)
@@ -180,13 +180,15 @@
     set_screen_state(int on)
     {
         //LOGI("*** set_screen_state %d", on);
+
+     if(!on)
+         return -EIO;

        initialize_fds();

        //LOGI("go_to_sleep eventTime=%lld now=%lld g_error=%s\n", eventTime,
            //      systemTime(), strerror(g_error));

-     if (g_error) return g_error;
+//     if (g_error) return g_error;

        char buf[32];
        int len;
@@ -197,6 +199,7 @@
        len = write(g_fds[REQUEST_STATE], buf, len);
        if(len < 0) {
            LOGE("Failed setting last user activity: g_error=%d\n", g_error);
+
+         return errno;
        }
        return 0;
    }
}
```

附錄V 建立 Android 執行環境的操作步驟

使用make menuconfig指令選取欲編譯的選項（各選項的詳細說明可以參考5.2.2），直接選擇儲存（我們預設的編譯選項即可建立Android的執行環境）並離開。我們預設的編譯選項如下：

- Linux kernel 2.6.25 with Android enabled
- Busybox 1.13.2
- Tiny root file system for Flash ROM
- Android demo file system for USB Flash
- JFFS2

將 Linux kernel 寫入 Flash ROM 的指令：

1. tftp a1100000 uImage
2. protect off 100000 47ffff
3. erase 100000 47ffff
4. cp.b a1100000 100000 200000

將 root file system 寫入 Flash ROM 的指令：

1. tftp a1480000 rootfs.jffs2
2. protect off 480000 97ffff
3. erase 480000 97ffff
4. cp.b a1480000 480000 500000

執行 Android 的指令：

1. cd /tmp
2. mkdir usb
3. mount /dev/sda1 ./usb
4. cd usb
5. chroot .
6. ./init

附錄W Android 相關的參考資料

- <http://android-opensocial.blogspot.com/2007/12/process-for-how-to-install-android-os.html>
- <http://androidzaurus.seesaa.net/article/87973048.html#comment>
- <http://benno.id.au/blog/>
- <http://benno.id.au/blog/2007/11/14/android-busybox>
- <http://blog.csdn.net/ydfok/archive/2008/03/26/2220708.aspx>
- <http://blog.jjgod.org/2007/11/>
- <http://discuz-android.blogspot.com/2008/01/customize-google-android-systemimg-for.html>
- http://embeddedlinux.movial.fi/Android_on_OMAP
- http://en.wikipedia.org/wiki/ARM_architecture
- http://en.wikipedia.org/wiki/Google_Android
- <http://euedge.com/blog/2007/12/06/google-android-runs-on-sharp-zaurus-sl-c760/>
- <http://feixf1974.javaeye.com/blog/192222>
- <http://groups.google.com/group/android-internals>
- <http://groups.google.com/group/android-porting>
- http://groups.google.lu/group/android-internals/browse_thread/thread/93570c41bce07f16?hl=en
- <http://hi.baidu.com/yty981/blog/item/83105229bd01d6fa98250a6c.html>
- <http://kerneltrap.org/node/4982>
- <http://kerneltrap.org/node/5091>
- <http://linux.idigitalnet.com/content/view/84/1/>

- <http://marc.info/?l=linux-omap&m=120470400929434&w=2>
- <http://memyselfandtaco.blogspot.com/>
- <http://momodalo.blogspot.com/2008/02/android-porting-experience.html>
- <http://nemustech.blogspot.com/2007/12/android-porting-to-real-target-hw.html>
- <http://pingooo.jaiku.com/presence/36324031>
- <http://rider51.wordpress.com/2008/05/13/simple-guide-for-porting-android-kernel/>
- <http://rider51.wordpress.com/2008/05/14/>
- <http://tree.celinuxforum.org/CelfPubWiki/Jamboree18AndroidDemo>
- <http://wiki.kldp.org/wiki.php/AndroidPortingOnRealTarget>
- http://wiki.tossug.org/%E5%BF%83%E5%BE%97%E5%88%86%E4%BA%AB2008#loving_git
- <http://www.androidin.com/android-886-1-1.html>
- <http://www.androidlab.cn/archiver/?tid-288.html>
- <http://www.blogged.com/blogs/rider-fairlady-blog.html>
- <http://www.cnblogs.com/herolf/archive/2008/01/20/1046456.html>
- <http://www.forwind.cn/2007/08/13/powerpc-linux2/>
- <http://www.geocities.com/sugopilius/netresource.htm>
- <http://www.ibm.com/developerworks/cn/linux/l-btloader/>
- <http://www.javaworld.com.tw/jute/post/view?bid=26&id=230240&sty=1&tpg=1&age=0>
- <http://www.linux.org.tw/news/feed>
- <http://www.mis.com.tw/GoogleAndroidSetup.pdf>

作者簡介

鍾文昌 〈cycdisk@gmail.com〉

1978.**.**

我從民國 93 年開始從事 Linux 及 Embedded Linux 的相關開發工作，接觸過 x86、MIPS 及 ARM platform，對 Linux kernel、Linux device driver、Shared Library、Application 等皆有所涉獵：

1. Linux kernel / device driver：

- Porting IR driver
- Customizing AC97 driver
- Audio driver

2. Shared Library：

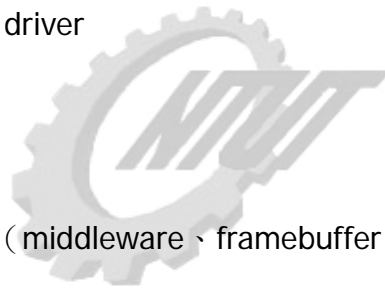
- SDK development (middleware、framebuffer、multicast)
- Firefox plug-in

3. Application：

- Socket programming
- Multi-thread (pthread)
- X programming
- Porting nano-X
- DVB-T (SDT、EIT...)
- Fdisk on DOS (used C / Assembly)

4. System Integration：

- Porting Android
- Porting QT



我很慶幸在過去遇到非常好的主管，不僅讓我在技術上有相當的精進，在做人處事方面也讓我受益良多，大大地改變我對人事物的看法。

若是有家扶中心或是弱勢團體需要電腦講師等幫助，我會盡可能地幫忙。



版權聲明：歡迎以任何形式自由散佈本論文，但請保留本論文的出處。