Tamkang University Software Engineering Group
淡 江 軟 體 工 程 實 驗 室
http://www.tkse.tku.edu.tw/

嵌入式系統概論與實作

Presenter：Ying-Hong Wang
E-mail：inhon@mail.tku.edu.tw
Date：2011/10/24

上課用書、參考用書暨相關規定

- **上課用書**
  - 圖書代理
  - 請尊重智慧財產權，勿非法影印使用教科書與參考書籍及使用盜版軟體
- **成績評定**
  - 出席:15% 作業+程式: 60% 期末報告: 25%
  - 出席成績每週點名一次，係用以考核出席與否，故不接受任何請假或補點，但每人每學期有3次免責，同樣的，全勤者學期成績外加三分。
  - 一般作業可以遲交，逾時，每24小時內扣10分，扣至0分為止
  - 隨堂作業不受理補交
  - 演講出席:每場加一分(需簽到簽退)
- **上課方式**
  - 板書為主、 投影片為輔
- **上課規定**
  - 手機改設震動或關機、不要私下講話
- **考試方式**
  - 無考試

金句名言分享

- 台灣首富郭台銘在其部落格上提到
  - 為金錢做事，容易累
  - 為理想做事，能耐風寒
  - 為興趣做事，則永不倦怠
- 成功的要素不僅止於有正確的策略，還要有
  - 快如閃電的速度
  - 狠如巨雷的執行力

對同學們的建議

- 在專業養成上
  - 奠定紮實的基礎
  - 養成終身學習的習慣
- 在人格養成上
  - 建立正確的態度
  - 處事三態：真誠、負責、合群

Outline

- Introduction to Embedded System Credits Program
- Introduction to Embedded Systems
- Understanding the ARM serious
- Famous Embedded OS
- Linux Overview
- Introduction to Linux Programming
- Guide to Programming on ARM serious
- Embedded Systems Development

Outline

- **Introduction to Embedded System Credits Program**
- Introduction to Embedded Systems
- Understanding the ARM serious
- Famous Embedded OS
- Linux Overview
- Introduction to Linux Programming
- Guide to Programming on ARM serious
- Embedded Systems Development

Introduction to Embedded System Credits Program

- 工學院『嵌入式系統』學分學程
  - 學程名稱：嵌入式系統
  - 學程類別：學分學程
  - 教學單位：資訊工程學系、電機工程學系
  - 學程業務由資訊工程學系承辦
  - 最低修業學分數：33學分
  - 參考網站：http://www.csie.tku.edu.tw
  - 修畢學程規定課程者，可申請由教務處發給「淡江大學嵌入式系統學分學程證明書」

TSEG Tamkang University Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

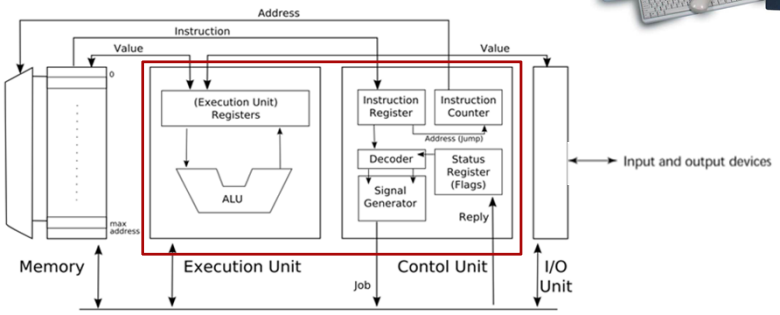
Introduction to Embedded System Credits Program

- 工學院『嵌入式系統』學分學程科目表
  - **基礎必修：**
    - 計算機組織 3學分
    - 二選一
    - {邏輯設計+數位系統設計} 4學分 (各2學分) 電機系開
    - {數位系統導論+數位系統實驗} 4學分 (3+1學分)資訊系開
  - **專業必選：**
    - 作業系統 4學分 資訊系開設
    - 微處理機概論 3學分 電機系開
    - 微處理機實驗 1學分 電機系開
    - 嵌入式系統概論與實作 3學分 工學院共同科開
    - 嵌入式系統程式設計 3學分 工學院共同科開

Outline

- Introduction to Embedded System Credits Program
- **Introduction to Embedded Systems**
- Understanding the ARM serious
- Famous Embedded OS
- Linux Overview
- Introduction to Linux Programming
- Guide to Programming on ARM serious
- Embedded Systems Development

Introduction to Embedded Systems



The Von Neumann Architecture

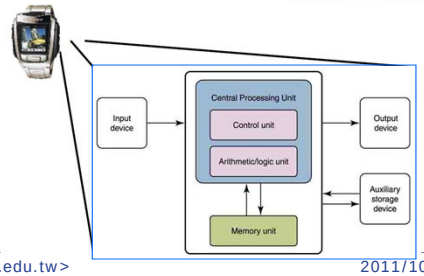
What is Embedded System ?

- 電腦，假裝自己不是電腦..... (Stephen A. Edwards)

What is Embedded System ?

- 為何電腦要嵌入？

- 基本架構仍然相同



Introduction to Embedded Systems

- 嵌入式系統定義
  - 英國電機工程師協會定義：嵌入式系統為控制、監視或輔助某個設備、機器或甚至工廠運作的裝置
    - 用來執行特定功能
    - 以微電腦與周邊構成核心
    - 嚴格的時序與穩定度要求
      ex. 在機器控制上，稍有不慎則可能失去控制，釀成災害。
    - 全自動操作循環
      ex. 斷電時的緊急處理，使復電後仍能回復原有的狀態。

嵌入式系統定義

- 所謂的嵌入式系統簡而言之是一種「執行部份特定功能」的系統，實作上並不限定技術範疇，只要能將特定的功能（function）「嵌入」到目標裝置（target device）裡，包含這些功能的整體系統（或平臺）即是「嵌入式系統」
- 嵌入式系統範疇
  - 硬體的晶片設計、電路與周邊設計，到軟體、韌體、系統整合，以及各式各樣的應用程式與服務提供，都是整個嵌入式系統領域包含的範疇

嵌入式系統概念

嵌入式系統特性

1.) 比較固定的硬體(不能擴充)

2.) 比較固定的軟體(不能擴充)

3.) 相對比較小的系統(與PC 比較)

4.) 採用的硬體大都SoC 化

5.) 綠色環保省電化

6.) 體積縮小化

7.) 產品降價快、淘汰快

8.) 數位化

9.) 具無線能力

嵌入式系統特性

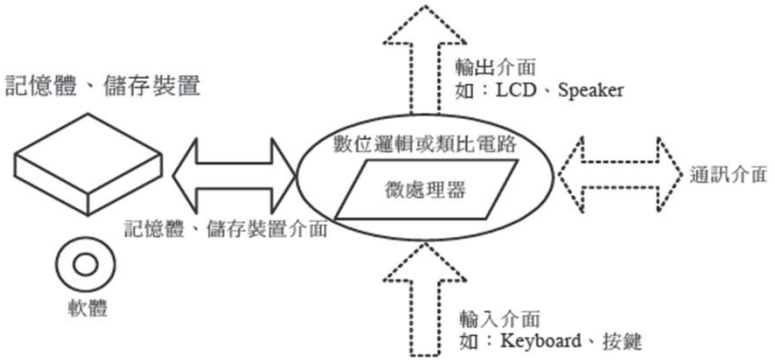
- 嵌入式系統為一種電腦軟體與硬體的綜合體，特別強調「量身定做」的原則，基於某一種特殊用途上，針對這項用途開發出截然不同的系統，是所謂的客制化(Customized)系統。
- 所以嵌入式系統是一種
  - 設計的目的，在於執行特定的功能
  - 以微電腦與周邊構成核心
  - 嚴格的時序與穩定度要求
  - 全自動操作循環

的電腦系統

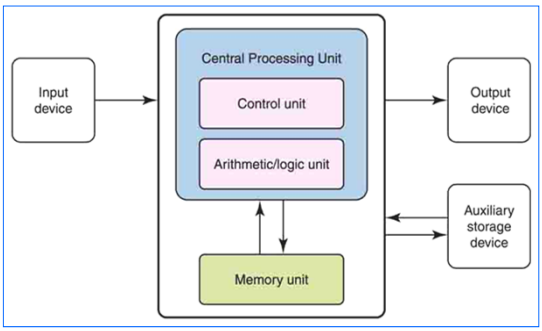
嵌入式系統產業

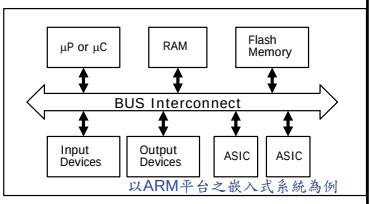
- 系統：嵌入式作業系統（RTOS）
  - VxWORKs、Nucleus 、Palm、Windows CE、Linux
- 整合式晶片：SoC
  - ARM、MIPS、Rambus、Mentor、Synopsys、Insilicon、DSP Group、VirageLogic、Artisan以及Parthus
- 應用軟體
  - 使用者端的應用軟體及伺服器端的整合軟體
- 服務
  - 日本NTT DoCoMo所發展的iMode服務
  - 3G
  - WEB 2.0

嵌入式系統組成組件



嵌入式系統組成組件





嵌入式系統組成組件

一、微處理器/微控制器：數位與類比電路， 它作為主控的核心。

二、輸入介面：它作為各種人機介面的輸入與信號的輸入， 以讓微處理器知道要做何事。

三、輸出介面：它作為各種人機介面的輸出與信號的輸出， 以讓微處理把運算結果呈現出來。

四、記憶體、儲存介面： 它作為微處理器所需的程式與資料來源處及程式與資料存放處。

五、通訊介面：它作為近處或遠處資料傳送或接收的窗口。

六、軟體/特殊用途之積體電路晶片(ASIC)等六部份。

嵌入式系統的應用

Embedded systems in everything

辦公設備、建築物設備、製造和流程控制、醫療、監視、維生設備、交通運輸、通訊業、銀行金融業、測試及診斷系統、其他

嵌入系統的未來發展

- **硬體:**
  1.) SoC 包入更多線路、元件
  2.) 雙核心及更多核心
  3.) 更省電
  4.) 更大及3D 影像處理能力
  5.) 另一種內含記憶體小型又更省電的32 bit 正在興起被應用(如 Cortex)

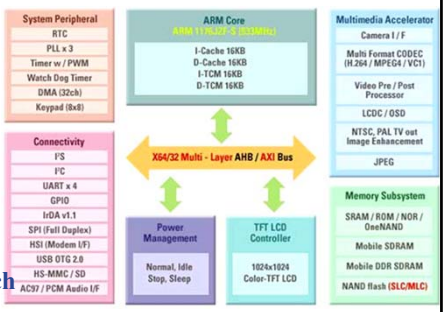
- **軟體:**
  1.) 更模組化的軟體，易被再使用
  2.) 多核心的作業系統及應用
  3.) 除錯更困難、更耗時

Application Case Studies

- **The LEGO Mindstorms® is a robotic building system consisting of**
  - **The NXT Intelligent Brick: the brain of the system**
  - **Sensors and servo motors**
  - **LEGO TECHNIC Elements**
  - **Programming software**
- **32-bit ARM7 microprocessor**
  – 256 Kbytes FLASH, 64 Kbytes RAM
- **Bluetooth wireless communication using NXT softw only.**
- **USB 2.0 port**
- **4 input ports, 6 wire digital platform**
- **3 output ports, 6 wire digital platform**
- **100 x 64 pixel LCD graphical display**
- **Sound channel with 8-bit resolution**
- **Stores multiple programs**

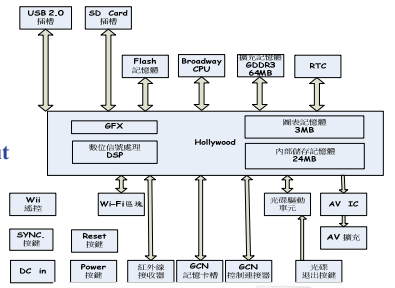
Application Case Studies

- iPhone®
- Went on sale in June 2007
- One Samsung ARM1176
  application processor core
- Memory capacity: 8 – 16 GB
- Display: 3.5 inch widescreen multi-touch
- Operating system: MAC OS X
- GSM: Quad-band
- Wireless data: WiFi, EDGE, Bluetooth
- Camera: 2 Mega pixel
- Audio and video playback
- Talk time: 8 hrs
- Standby time: 250 hrs



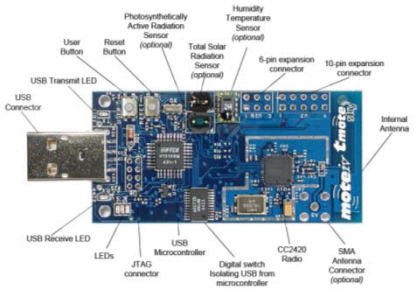
Application Case Studies

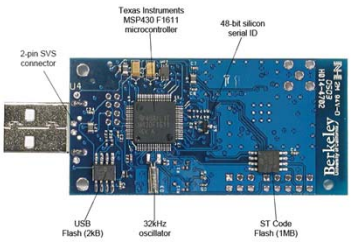
- Wii
- CPU – IBM "Broadway"
  - Runs at 729 MHz
  - Similar to the chip found in the GameCube, but almost twice as fast
- ATI "Hollywood" processor handles the graphics
- Runs at 243 MHz
  - 24 MB of internal main memory
  - 64 MB GDDR3 SDRAM
- Up to four Wii-motes
  – Connected via Bluetooth
- SD memory slot
- GameCube memory slot
- 512 MB built in NAND flash memory



Application Case Studies

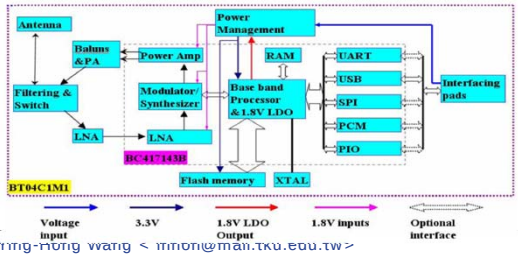
- **Tmote Sky Sensor**
  - Widely proven platform for wireless sensor systems deployments
  - Tmote Sky offers a number of integrated peripherals including a 12-bit ADC and DAC, Timer, I2C, SPI, and UART bus protocols, and a performance boosting DMA controller
  - Tmote Sky is the wireless sensor module of choice for large-scale, high data-rate sensor network applications requiring ultra low-power, high-reliability and ease of development





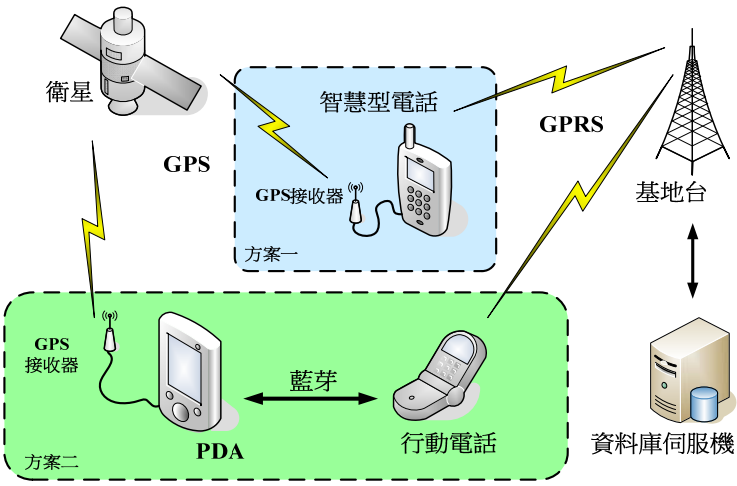
Application Case Studies

- **BTM3015EC4C5-00 Bluetooth Module**
  - Bluetooth Specification V 1.2/2.0 compliant
  - Enhance data rate(EDR) V0.9 compliant
  - Maximum transmit Power up to +14 dBm (Class 1)
  - Low power consumption for running on 1.8V core
  - Full Bluetooth data rate over UART and USB
  - On-board Flash memory (8Mbits)
  - 12 Programmable I/O lines



Application Case Studies

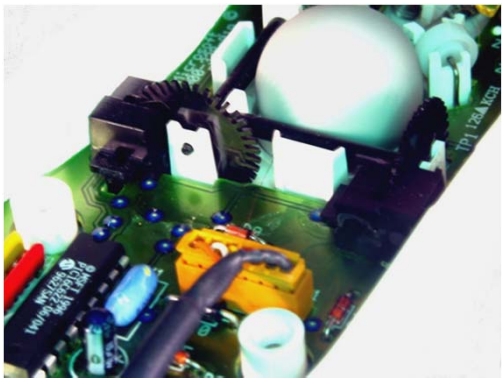
- GPRS/GPS導航器功能



Examples of Embedded Systems

- Product: Sonicare Elite toothbrush
- Microprocessor: 8-bit
- Has a programmable speed control, timer, and charge gauge

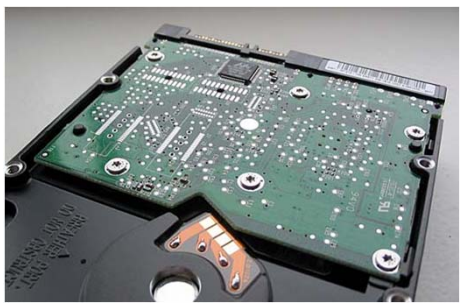
Examples of Embedded Systems



Inside view of a Microsoft Mouse

- Product: Any PC Mouse, Keyboard, or USB Device
- Microprocessor: 8-bit Microcontroller

Examples of Embedded Systems

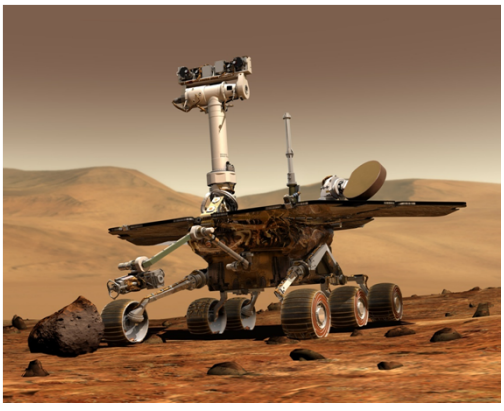


- Product: Any Disk Drive
- Microprocessor: Dual 32-bit Marvel ARM SOC & mixed signal DSP

Examples of Embedded Systems

- Product: Vendo Vue 40 vending machine
- Microprocessor: Two 16-bit Hitachi H8/300H Processors
- A robot hand dispenses items

Examples of Embedded Systems



- Product: NASA's Twin Mars Rovers
- Microprocessor: Radiation Hardened 20Mhz PowerPC
- Commercial Real-time OS
- Software and OS was developed during multi-year flight to Mars and downloaded using a radio link

Examples of Embedded Systems

- Product: Sony Aibo ERS-7 Robotic Dog
- Microprocessor: 64-bit MIPS R7000
- OS: Aperios - Sony's Real Time OS

Examples of Embedded Systems

- Product: Radiant Systems Point-of-Sale (POS) Terminal
- Microprocessor: Intel X86 Celeron
- OS: Windows XP Embedded
- Look for one next time you order fast food!

Examples of Embedded Systems

- Product: Microsoft's Smart Personal Object Technology (SPOT) watch
- Microprocessor: 32-bit ARM with FM Radio Chip
- Downloads News using extra bandwidth on FM radio stations in major cities

Examples of Embedded Systems

- Product: Motorola Q Pocket PC Phone
- Microprocessor: TI OMAP (ARM+DSP)
- OS: Windows Mobile 5.0 (Windows CE OS)

Examples of Embedded Systems

- Product: Atronic Slot Machine
- Microprocessor: X86
- OS: Windows CE
- Slots are networked to a casino's back office computer system. State Gaming agencies audit code and payoff odds

Examples of Embedded Systems

- Product: Dresser Wayne Ovation iX Gas Pump
- Microprocessor: Marvel Xscale (ARM)
- OS: Windows CE
- Displays video ads & is networked to a gas station's back office computer system. Also has remote maintenance features

Examples of Embedded Systems

- Product: Bernina Artista 200 Sewing Machine
- Microprocessor: Marvel StrongARM
- OS: Windows CE
- Can download new images from Internet and sew them

More Examples

Aircraft & Military Systems	Aircraft autopilots, avionics and navigation systems, automatic landing systems, guidance systems, engine controls.
Biomedical	XRAY, MRI, and Ultrasound imaging systems, patient monitors, heart pacers.
Cars	Engine control, anti-lock braking systems, traction control systems, air bag controls, heating and air conditioning controls, GPS mapping, Satellite Radio, On-board Diagnostics.
Communications	Communication Satellites, network routers, switches, hubs.

More Examples

Computer I/O devices	Keyboards, mouse, printers, scanners, displays, modems, hard disk drives, DVD drives, graphics cards, USB devices.
Electronic Instrumentation	Oscilloscopes, voltmeters, signal generators, logic analyzers.
Home Electronics	Microwave ovens, dishwashers, DVD players, televisions, stereos, security systems, cameras, TVs, clock radios, answering machines, satellite or cable box
Industrial Equipment	Elevator controls, surveillance systems, robots, CNC machines, Programmable Logic Controllers, industrial automation and control systems.

More Examples

Office Machines	FAX machines, copiers, telephones, calculators, cash registers.
Personal Devices	Cell phones, portable MP3 players, Video players, Personal Digital Assistants (PDAs), electronic wrist watches, handheld video games, digital cameras, GPS systems.
Robots	Industrial robots, autonomous vehicles, space exploration robots (i.e. Mars robots)
Toys	Video Game systems, “Aibo”, "Furby“, and “Elmo” type robot toys.

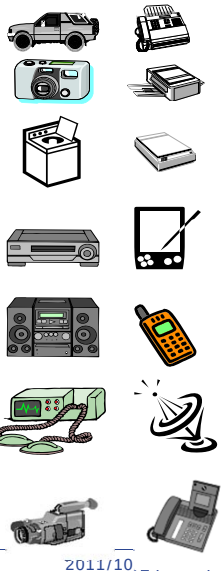
A "short list" of embedded systems

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems
Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR's, DVD players
Video game consoles
Video phones
Washers and dryers

Computers are in here...

and here...

and even here...



And the list goes on and on

Three Kinds of Products

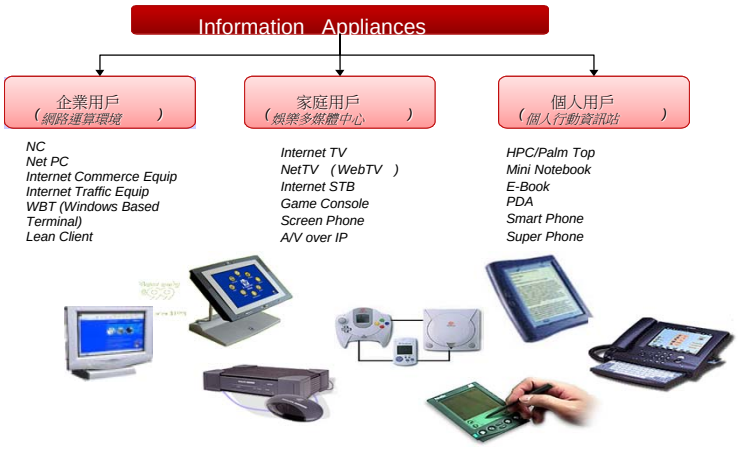
- **個人型/行動型(personal/mobile)資訊家電產品**
  - Ex：電子書、手持裝置

- **家庭式/娛樂式/視聽式(Home/Entertainment &Audio/Video)**
  - Ex：網路電視、多媒體視訊設備

- **企業型/網路型(Enterprise/Networking)**
  - Ex：嵌入式伺服器、精簡型終端設備

Three Kinds of Products

- IA產業之各種產品關聯



資料來源：工研院經濟中心、經濟部IT IS計畫

What I can do ?

- Skills and knowledge required for design of embedded systems
  - Hardware
    - Microprocessors, digital logic, FPGAs, computer architecture/design, networking, etc.
  - Software
    - Communication Protocol Design, Communication Interface and Control, Assembly and high-level programming, operating systems, software engineering, etc.
  - Firmware
    - In embedded systems, the software typically resides in firmware, such as a flash memory or read-only memory (ROM) chip, in contrast to a general-purpose computer that loads its programs into random access memory (RAM) each time.

What I can do ?

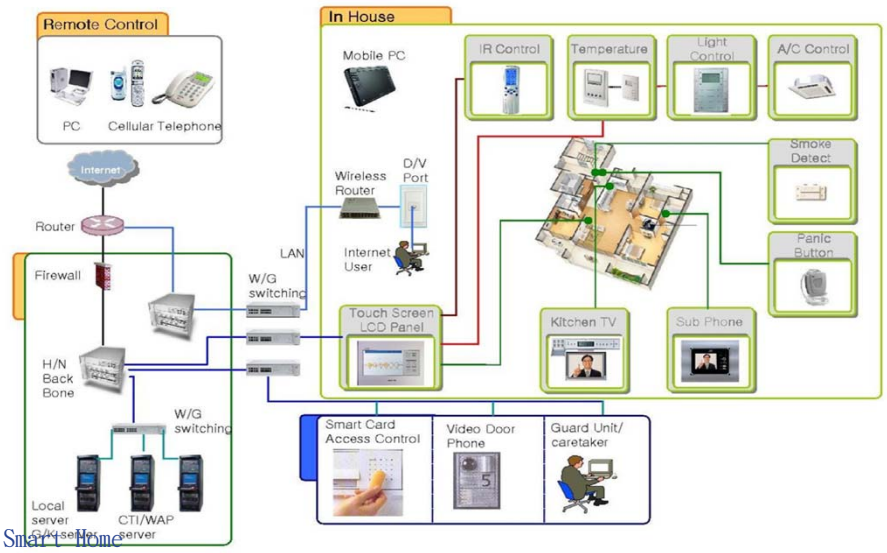
- **嵌入式系統設計能力**
  - 硬體的晶片設計、通訊元件、電路與周邊設計， 到軟體、韌體(Firmware)’作業系統與系統整合，以及各式各樣的通訊、應用程式與服務提供，都是整個嵌入式系統領域包含的範疇
  - 一個軟體工程師應該具備的能力
    - 系統整合能力
    - 程式語言能力
    - 可視度設計能力
    - 硬體界面設計能力
    - 掌握市場應用導向 (無線通訊與控制之整合)

System integrity capacity

- 嵌入式系統屬於計算機系統的一支，計算機架構、組織與作業系統是基本的基礎，在軟體包括RTOS、DSP與人工智慧；硬體技術則包含電子電路整合電路設計技術

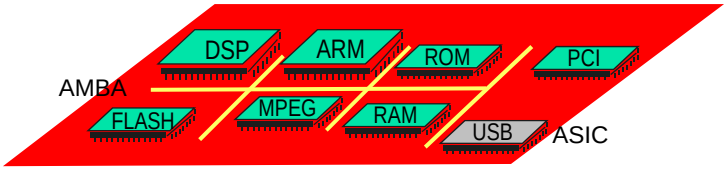
Smart Home

System integrity capacity



Smart Home

Hardware Design -- Chip



- **Application specific integrated circuit (ASIC晶片)**

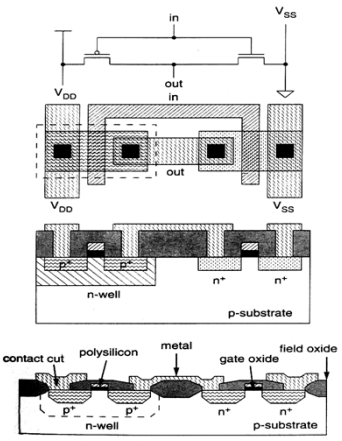
  such as USB, MPEG, ….

- **Full custom design vs. semi custom (cell-based) design**
- **The basic design flow for digital cell-based ASIC**
  - Describe circuits with hardware description language (HDL硬體描述語言) VHDL and Verilog
  - Synthesis (合成) the circuits ….
  - Need the concepts of digital logic/system

Hardware Design -- Chip

- **Full Custom Design (全客戶式設計)**
  - Digital circuits requiring custom optimization (smaller system)
  - Analog circuits
  - Long design cycle (transistors and wires)
  - No CPLD or FPGA solutions

CMOS Inverter in out



done by chip designer

masking

done by TSMC, UMC

Packing, Testing

Hardware Design -- Chip

- **Semi Custom Design (半客戶式設計)**

a. Product specification
b. Modeling with HDL
c. Synthesis (by using suitable standard cell)
d. Simulation and verification
e. Physical placement and layout

-- implemented with suitable tools

f. Tape-out (real chip) -- implemented by suitable Fab companies
g. Testing -- implemented by suitable tools and mechanisms

more flexible, shorter design cycle, suitable for smaller production

Two different solutions :

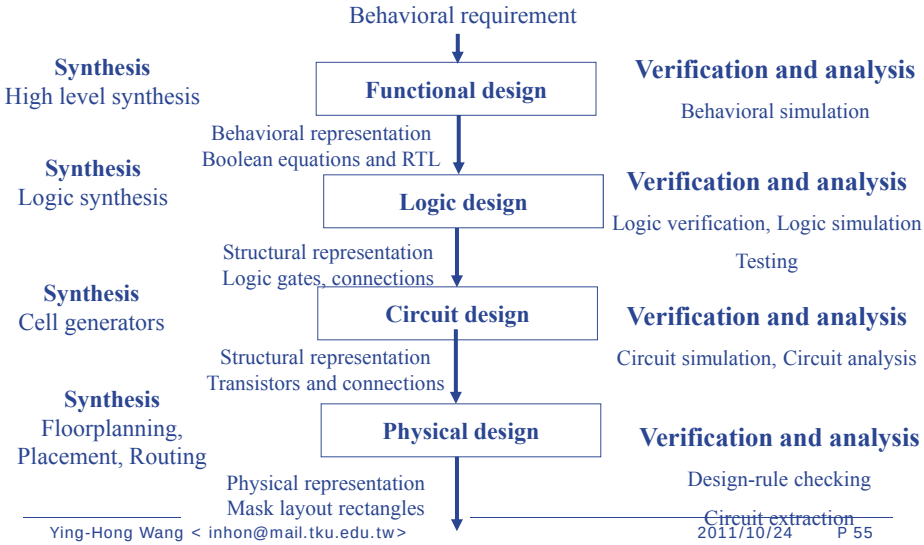
PLD → FPGA or CPLD — Xilinx, Altera

Standard cell → Real ASIC chip — Fab (TSMC, UMC, ..)

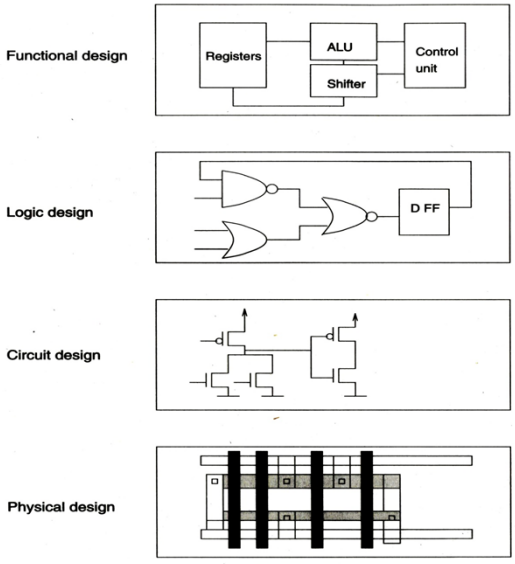
less flexible, long design cycle, larger-scale production to reduce price

Hardware Design -- Chip

- **Synthesis Flow of Semi Custom design**

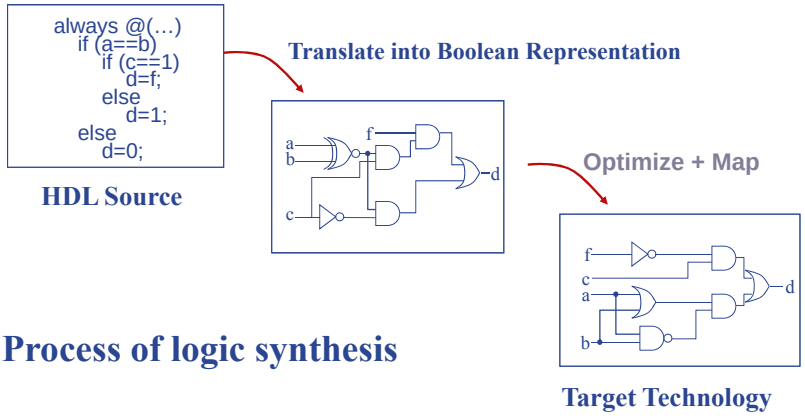


Hardware Design -- Chip



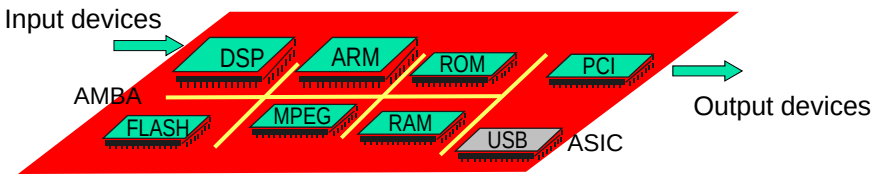
Hardware Design -- Chip

- *Synthesis = Translation+Optimization+Mapping*



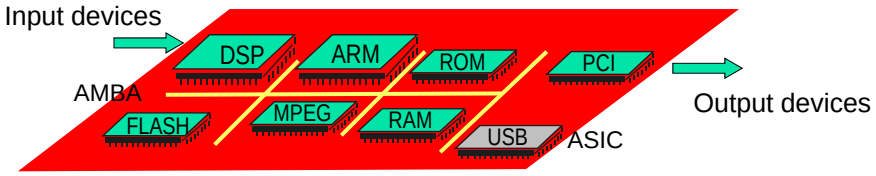
Process of logic synthesis

Firmware Design



- **Devices drivers for I/O devices, extended devices, transmission interface**
- **Assembly codes and C codes for some dedicated CPUs (ARM, 8086,..)**
  - Architectures and instruction sets of different CPUs, DMA,…

Software Design

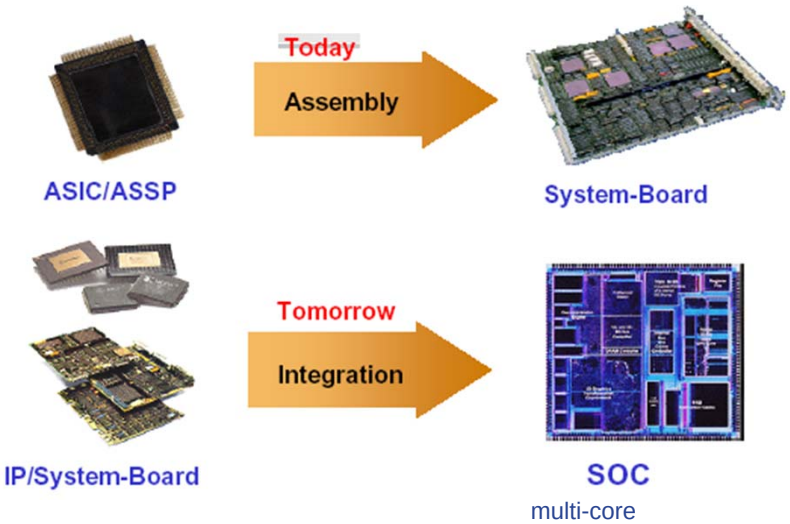


- **Embedded OS: WinCE, Palm OS, uC/OS, Linux, JAVA**
  - ➔Real time OS (time) ➔ as small as possible (memory)

    ➔Distributed embedded system (+ fault tolerance)
- **Application Software:**
  - wireless communication, networking, multimedia,

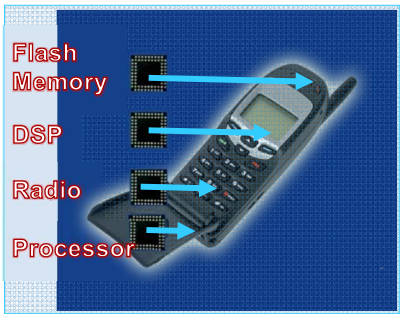
    health, convenience, Web, ….
  - Porting a customized embedded system to different machines is very difficult (need large modification)

Future



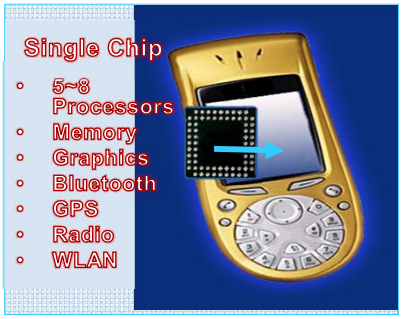
Example: Mobile Phone

Yesterday



- Voice only; 2 processors
- 4 year product life cycle
- Short talk time

Today

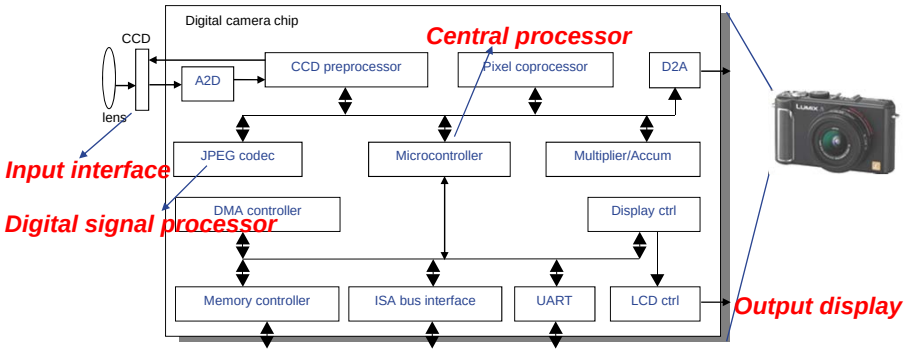


- Voice, data, video, SMS
- <12 month product life cycle
- Lower power; longer talk time

Source: EI-SONICS

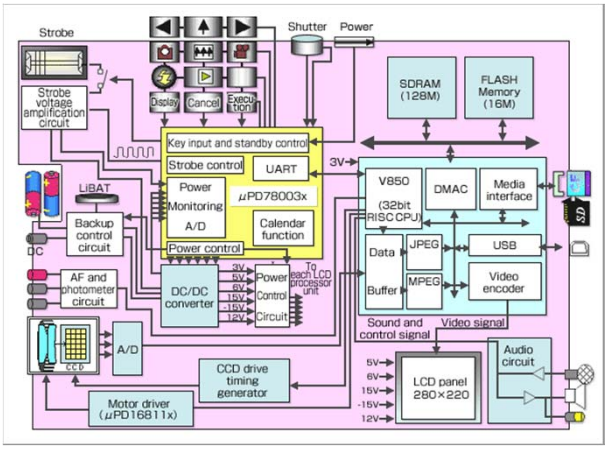
Example: Digital camera

- A digital camera



- Single-functioned -- always a digital camera
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time -- only to a small extent

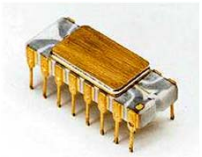
Example: Digital camera



- There are cases in which digital cameras realize all of their functions through the main chip and do not employ a subchip

嵌入式系統的發展歷史第一階段

- 嵌入式系統的興起是在1971年，由Intel公司推出有史以來第一顆四位元微處理器4004



- 通常是以單晶片為核心，應用於一些工業控制系統中。這種嵌入式系統，通常沒有作業系統的支援，通過組合語言程式執行系統的功能，一般具有與監測、服務以及控制功能。
- 特點
  - 系統結構和功能都相對簡單成本較低，這種嵌入式系統屬於低階應用領域
  - 此種嵌入式系統已經不能適應現代工業控制和資訊家電等需求，目前已逐步退出產業界

嵌入式系統的發展歷史第二階段

- 以嵌入式CPU為基礎和嵌入式作業系統為核心的嵌入式系統
- 隨著嵌入式微處理器的發展，以及嵌入式作業系統設計開發水準的提高，造成這個階段的嵌入式作業系統性能亦不斷提高。
- 嵌入式作業系統能運行於不同類型的微處理器上，具有
  - OS 核心小
  - 特殊用途之效率高
  - 高度的模組化
  - 擴展性

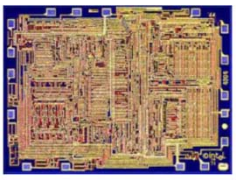
嵌入式系統的發展歷史第三階段

- 基於Internet為目標之嵌入式系統，目前正在迅速發展。進入90年代後，Internet的應用迅速普及，廣泛深入到社會、經濟、軍事、交通、通信等相關行業，消費電子、電腦與通信一體化的趨勢日趨明顯。嵌入式技術再度成為一個研究的熱門主題。
- **第三階段之嵌入式系統可提供**
  - 1. 多工處理
  - 2. 多程序處理
  - 3. 多種設備與輸出入介面支援
  - 4. 網路/無線通訊功能
  - 5. 圖形化視窗
  - 6. 使用者介面等功能
  - 7. 提供開發者大量的應用程式介面，開發應用程式相對簡單
  - 8. 嵌入式作業系統

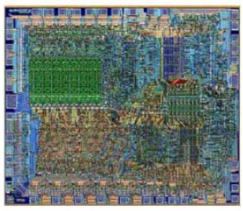
Various Microprocessors

- **常見的微處理器處理的資料位元**
  - 8 bits : Intel的8080, 8051 和Motorola的6800
    - ex：馬達控制器、電視遊樂器、冷氣機、電腦顯示器、傳真機、呼叫器、家電用品、電話答錄機…
  - 16 bits: Intel的8086 , 80286 和Motorola 的68000 ,
    - ex ：大哥大、攝影機、錄放影機、各式多媒體應用…
  - 32 bits: Intel的80386 , PXA250和Motorola的68020
    - ex：數據機、掌上型電腦、路由器、工作站、雷射印表機、彩色傳真機、數位相機、衛星定位系統、網路家電…
  - 64 bits: Alpha CPU
    - ex：高階工作站、新型電視遊樂器、多媒體應用…

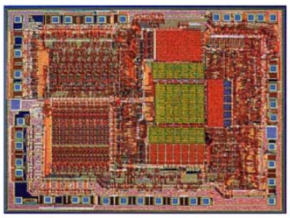
Cell-structure



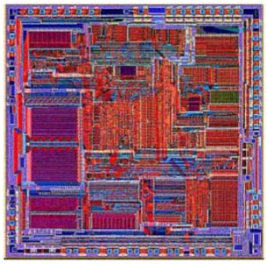
Intel 4004 (‘71)



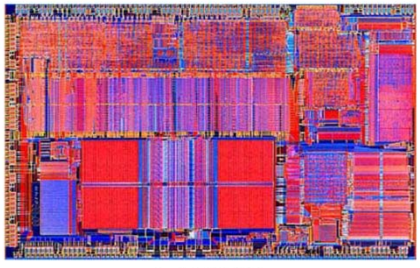
Intel 8080



Intel 8085



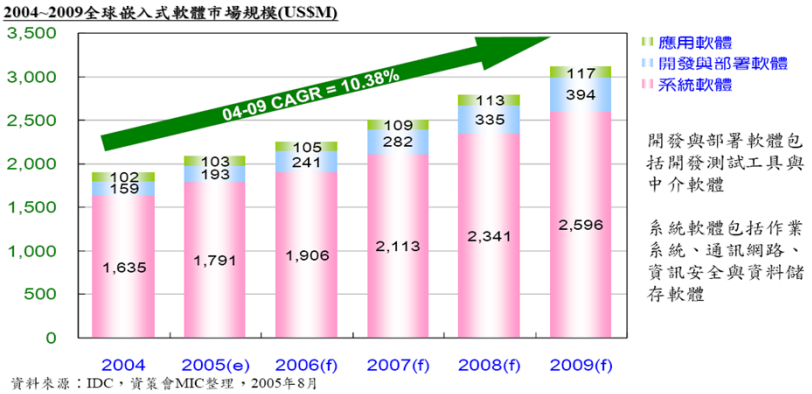
Intel 8286



Intel 8486

The Industrial Tendency

- 2009年全球嵌入式軟體市場將突破30億美元

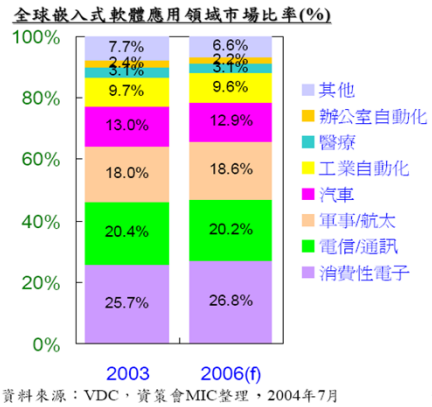


資料來源：IDC，資策會MIC整理，2005年8月

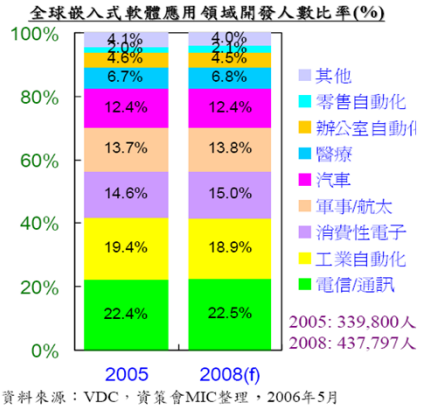
- 資通訊設備製造穩定成長，消費性電子與網通之需求在全球持續擴大，硬體製造大廠推展新產品，帶動全球嵌入式軟體市場

The Industrial Tendency

- 電信與通訊為嵌入式軟體之最大應用領域



資料來源：VDC，資策會MIC整理，2004年7月



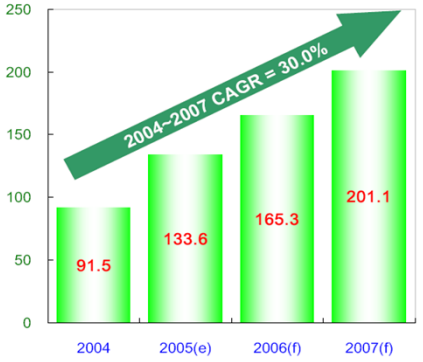
資料來源：VDC，資策會MIC整理，2006年5月

- 隨著VoIP與WLAN設備與服務之市場成長，電信與通訊大廠持續採用嵌入式作業系統與開發工具，並投入發展新一代應用軟體
- 消費性電子隨著電信、電腦、電視與廣播四大網路之逐步整合，轉變為智慧型嵌入式資通訊終端裝置，應用範圍持續擴大，促使嵌入式軟體需求與日俱增

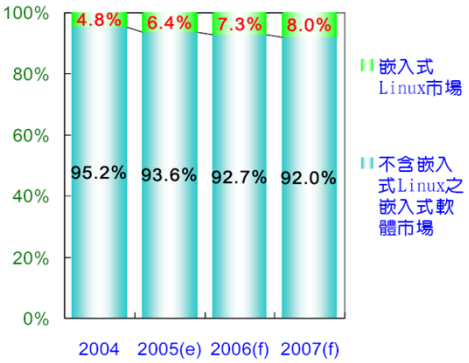
The Industrial Tendency

- 嵌入式Linux市場佔有率持續攀升

2004~2007年全球嵌入式Linux市場規模(US$M)



2004~2007年嵌入式Linux佔全球嵌入式軟體市場比率(%)



資料來源：IDC、VDC，資策會MIC整理，2005年11月

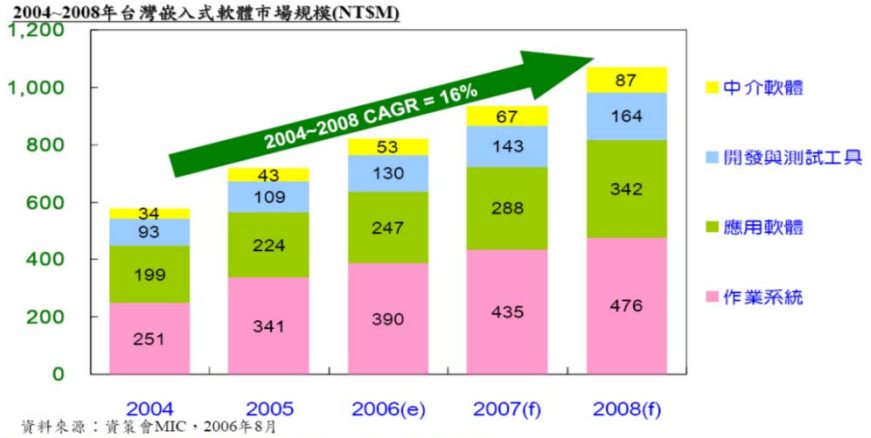
- 硬體廠商持續支持產品搭載嵌入式Linux方案，以降低客戶在軟體部分的支出比率
- 消費性電子、電信、通訊與工業自動化等應用領域採用嵌入式Linux之硬體廠商持續增加
- 通訊堆疊完整性是嵌入式Linux重要優勢

TSEG Tamkang University Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

The Industrial Tendency

- 2008年台灣嵌入式軟體市場將超過10億台幣

2004~2008年台灣嵌入式軟體市場規模(NT$M)



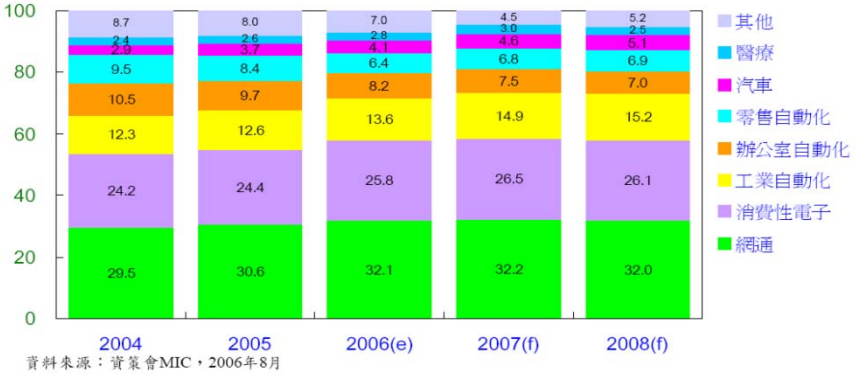
資料來源：資策會MIC，2006年8月

- 網通與工業自動化為台灣嵌入式作業系統主要需求
- 消費性電子與辦公室自動化為嵌入式應用軟體之主要成長因素
- Set Top Box 與 Thin Client等嵌入式資訊設備帶動中介軟體市場發展

The Industrial Tendency

- 網通為台灣嵌入式軟體主要應用領域

台灣嵌入式軟體應用領域市場比率(%)



資料來源：資策會MIC，2006年8月

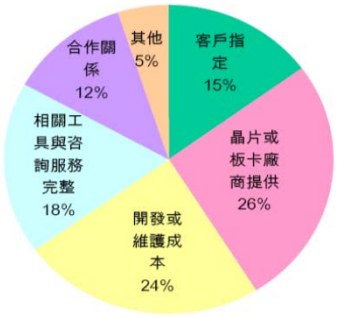
- Switch、Router、WLAN NIC、WLAN AP、DSL CPE與Cable Modem等是網通主要應用產品
- DVD錄放影機、PDA、STB、數位相機與電子字典等是消費性電子主要應用產品
- 工業電腦、Server Appliance與POS設備分別為工業自動化、辦公室自動化與零售自動化之主要應用產品

The Industrial Tendency

- 美日廠商為台灣嵌入式軟體市場要角

2006年台灣嵌入式軟體來源(%)

2006年台灣嵌入式軟體選擇考量(%)



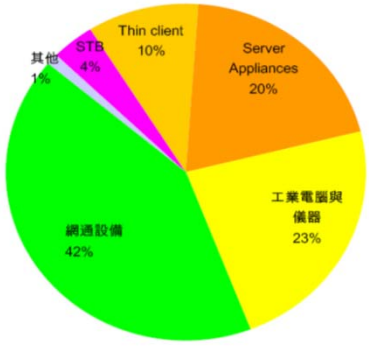
資料來源：資策會MIC，2006年8月

- 北美專屬作業系統仍為台灣市場主流，主要以專業嵌入式軟體廠商行銷台灣製造大廠
- 日本主要係透過硬體原廠以代工委託之方式綑綁嵌入式軟體方案

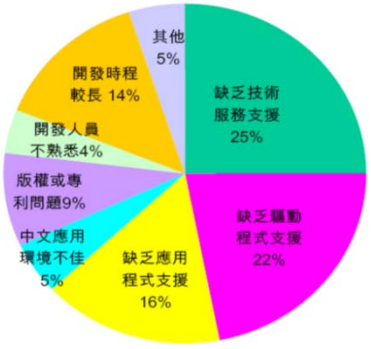
The Industrial Tendency

- 台灣嵌入式Linux之應用領域持續擴大

2006年台灣嵌入式Linux之硬體應用產品類別(%)



未採用嵌入式Linux之主要原因(%)



資料來源：資策會MIC，2006年8月

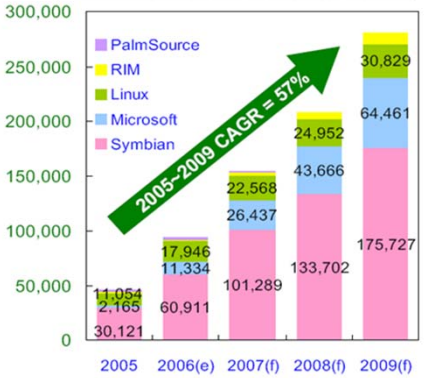
- 台灣嵌入式Linux之硬體應用產品類別持續擴大，目前已包括網通(主要為DSL與Router)、工業電腦與儀器、精簡伺服器(Server Appliance，含防火牆、郵件伺服器與NAS網路儲存裝置)、精簡終端(Thin Client)、STB、GPS，以及PMP等
- 缺乏技術、服務與程式支援，以及預期較長的開發時程是嵌入式Linux市場發展主要弱點

TSEG Tamkang University Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

The Industrial Tendency

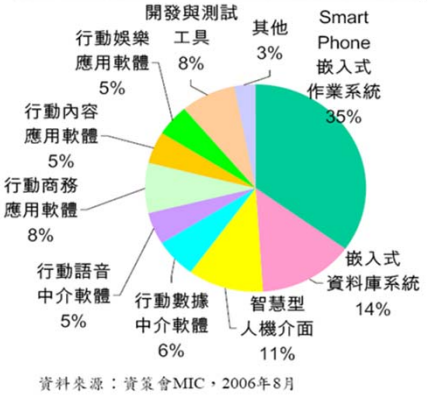
- 手機應用智慧化引領嵌入式行動軟體需求

2005~2009年全球智慧型手機出貨量(千台)



資料來源：Gartner，資策會MIC整理，2006年3月

手機應用智慧化相關之嵌入式軟體市場比率(%)

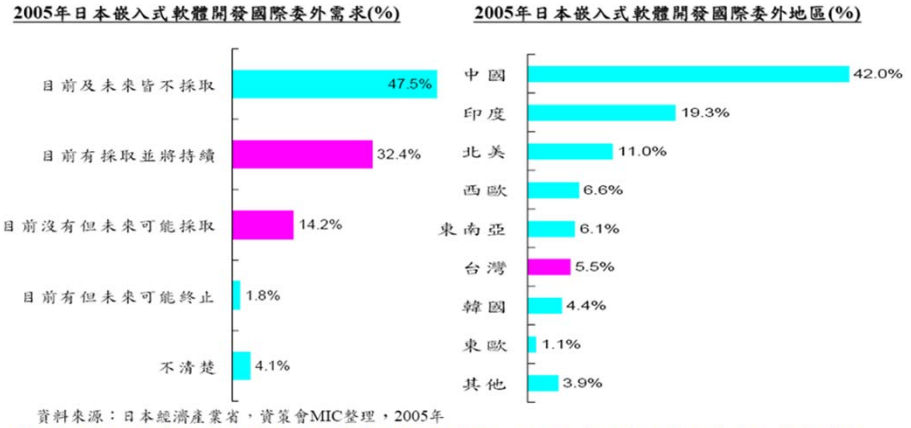


資料來源：資策會MIC，2006年8月

- 隨著手機由mobile terminal device走向mobile computing device的應用模式，智慧型、精緻化之嵌入式人機介面(Embedded GUI)以及精簡型之嵌入式資料庫系統(Embedded DBMS)軟體需求逐步擴大

The Industrial Tendency

- 嵌入式軟體國際委外商機競爭激烈

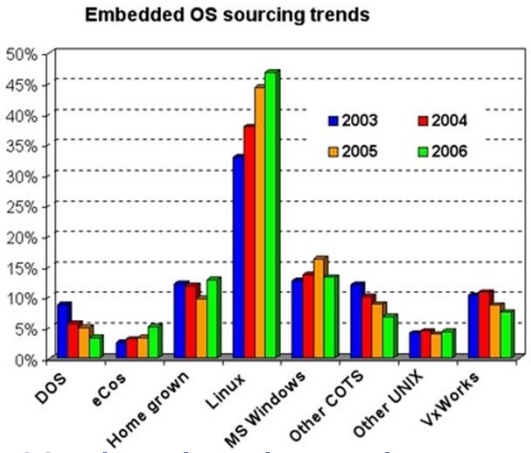


資料來源：日本經濟產業省，資策會MIC整理，2005年

- 2005年日本嵌入式軟體開發之需求人數為25萬人，而日本國內只有17萬嵌入式軟體開發人員，短缺8萬人，造成國際委外之大量需求
- 隨著全球化市場趨勢之發展，未來嵌入式軟體開發之國際委外將更為盛行，但業務競爭也益形激烈

The Industrial Tendency

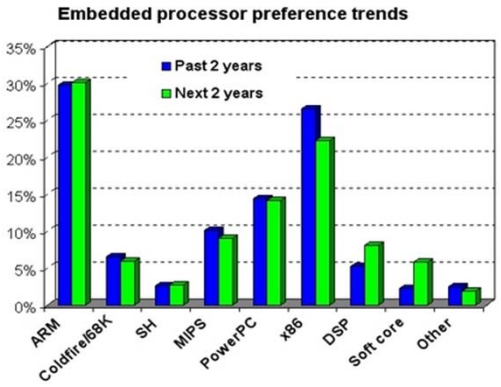
- Embedded Linux市場狀況(一)



Which OSes have been in your (company's) embedded designs during the past two years?

The Industrial Tendency

- Embedded Linux市場狀況(二)



- What CPU(s) have been in your (company's) embedded designs during the past two years? The next two?

The Industrial Tendency

- 長期人才供需推估

圖 5-1 94 至 104 年學士程度科技人力供需比較

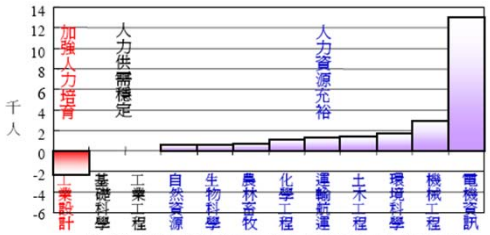
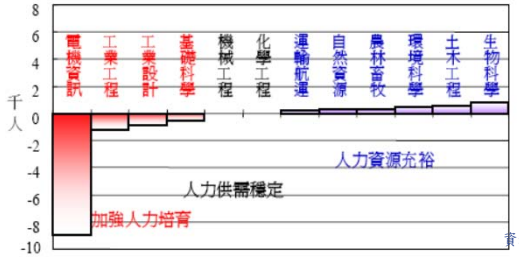
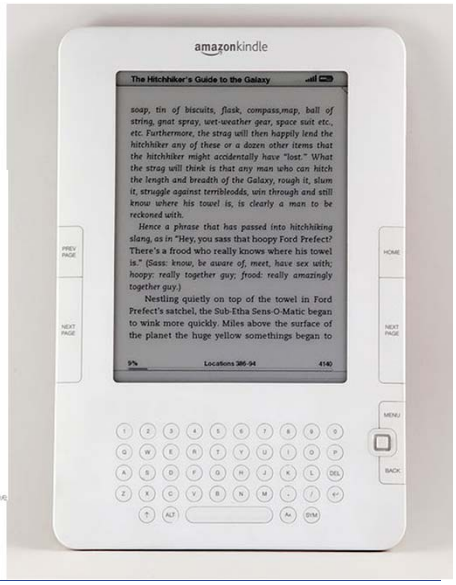


圖 5-2 94 至 104 年碩士以上程度科技人力供需比較



Amazon Kindle



Amazon Kindle

- Kindle的屏幕使用了叫作electronic paper的技術，使閱讀如同紙張一般

- Kindle的尺寸為7.5″ x 5.3″ x 0.7″（19x13.5x1.8cm）
- 重量10.3 ounces（290g）
- 6英寸的螢幕
- 分辨率600 x 800 pixel 167 ppi
- 有一個QWERTY鍵盤，讓輸入功能變得方便

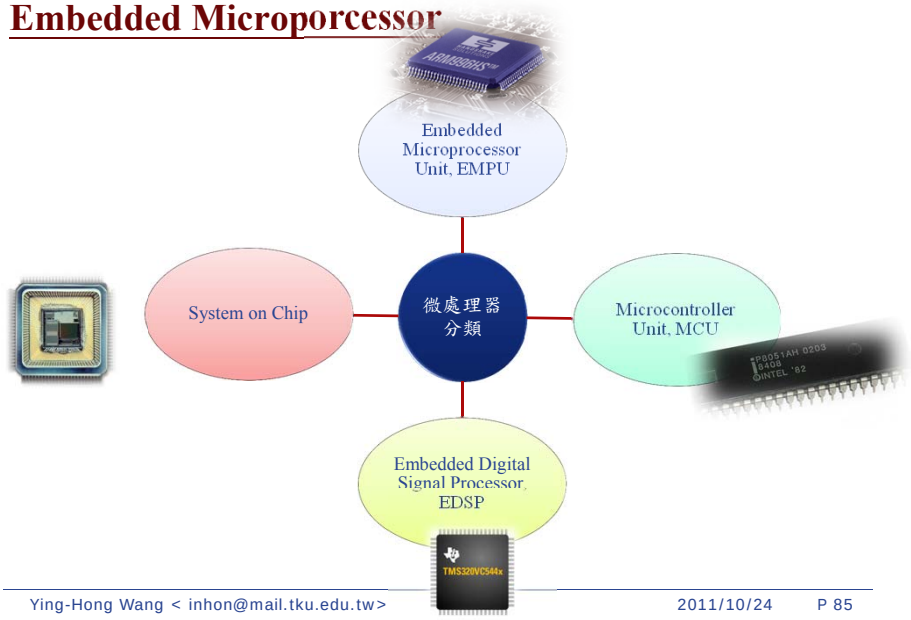
Amazon Kindle

- 開啟無線網路可使用2天，關閉網絡電池可續航一周
- 人機設計的外形和按鍵的排布，讓人如同拿一本書一樣
- 人們在長時間的閱讀總不斷的變換姿勢，兩側都有長條的翻頁按鍵，不僅方便不同姿勢下的閱讀，對於左右手習慣也可照顧到
- 目前Amazon提供逾9萬種電子書供用戶下載，還可以訂閱報紙雜誌
- 元太科技是國內首家拿下亞馬遜書店Kindle電子紙閱讀器訂單的龍頭業者

Outline

- Introduction to Embedded Systems
- **Understanding the ARM serious**
- Famous Embedded OS
- Linux Overview
- Introduction to Linux Programming
- Guide to Programming on ARM serious
- Embedded Systems Development

Embedded Microporcessor



Embedded Microporcessor

- **微處理器分類**
  - **嵌入式微處理器** Embedded Microprocessor Unit, EMPU
    - 微處理器內部僅包含單純的中央處理器單元，稱為一般用途型微處理器，類似於一般電腦中的CPU
    - 專門設計的電路板 。
    - 只保留和嵌入式應用有關的母板功能 。
    - 增強工作溫度、抗電磁干擾、可靠性等方面。
    - 功能與和工業控制電腦相似。
    - 嵌入式處理器目前主要有Am186/88、386EX、SC-400、Power PC、68000、MIPS、ARM系列等。
    - 目前最廣受市場歡迎的嵌入式處理器就是由ARM公司出品的ARM系列處理器

Embedded Microporcessor

- **微處理器分類**
  - **嵌入式微控制器** Microcontroller Unit, MCU
    - 以某一種微處理器內核為核心，晶片內部整合ROM/EPROM、RAM、匯流排、匯流排邏輯、定時/計數器、WatchDog、I/O、串列埠、脈寬調製輸出、A/D、D/A、Flash RAM、EEPROM等各種必要功能和介面，將其中央處理器、ROM、RAM及I/O等等裝置做到同一顆晶片上，這種微處理器稱為單晶片控制器（Single Chip Microcontroller）
    - 微控制器的最大特點是單晶片化，體積大大減小，從而使功耗和成本下降、可靠性提高。
    - 另外還有許多半通用系列如：支援USB介面的MCU 8XC930/931、C540、C541；支援I2C、CAN-Bus、LCD及眾多專用MCU和相容系列。
    - 代表性的系列包括8051、P51XA、MCS-251、MCS-96/196/296、C166/167、MC68HC05/11/12/16、68300、84等。目前MCU占嵌入式系統約70%的市場。

Embedded Microporcessor

- **微處理器分類**
  - **嵌入式數位訊號處理器** Embedded Digital Signal Processor, EDSP
    - DSP處理器對系統結構和指令進行了特殊設計，使其適合於執行DSP演算法，編譯效率較高，指令執行速度也較高。
    - 在數位濾波、FFT、頻譜分析等方面DSP演算法正在大量進入嵌入式領域，DSP應用正從在一般單晶片中以普通指令實現DSP功能，過渡到採用嵌入式DSP處理器。
    - 代表性的產品是Texas Instruments的 TMS320系列和Motorola的DSP56000系列。
    - 數位訊號處理器（Digital Signal Processor，DSP），最著名的是美國德州儀器公司CX家族系列
    - 兩個發展方向，
      - 經過單晶片化、電磁相容性 (EMC) 改造、增加晶片上週邊設備
      - 增加DSP輔助運算器
    - REAL DSP處理器，特點是具備雙哈佛架構 (Harvard architecture) 和雙乘/累加單元

Embedded Microporcessor

- 微處理器分類
  - 系統晶片 System on Chip
    - 在一個矽晶片上實現一個複雜的系統。
    - 整個嵌入式系統大部分均可整合到一塊或幾塊晶片中。
    - 應用系統電路板將變得很簡潔 。
    - SoC可以分為通用和專用兩類。
    - 通用系列包括Infineon (Siemens)的TriCore，Motorola的M-Core，某些ARM系列元件，Echelon和Motorola聯合研製的Neuron晶片等。
    - 專用SoC一般專用於某個或某類系統中。有代表性的產品是Philips的Smart XA

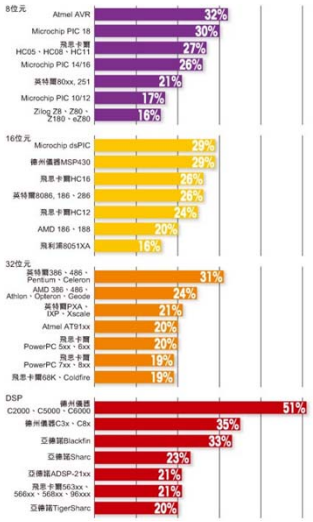
Embedded Microporcessor

- **There are many different CPU architectures used in embedded designs such as ARM, MIPS, Coldfire/68K, PowerPC, X86, PIC, 8051, ATMEL AVR etc…**
- **For more complex applications, 8/16-bit microprocessors are no longer suitable for the system because of the requirements of performance and functionalities.**
- **RISC (Reduced Instruction Set Computer)**
  - In the mid-1980s to early-1990s, a crop of new high-performance RISC microprocessors appeared
  - Some companies have attacked niches in the market, notably ARM, originally intended for home computer use but since focused at the embedded processor market
  - Today RISC designs based on the MIPS, ARM, PowerPC core power the vast majority of computing devices

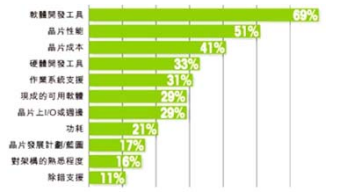
Embedded Microporcessor

- **Modern Requirements for embedded 32-bit RISC**
  - Internet support
    - TCP/IP stack (This is hard for 8-bit CPU to achieve)
    - WiFi, WiMAX
  - Multithread support
    - Operating System (Linux, Windows, …)
  - Large Memory, File-system
    - Storage (GIS data, DSC, …)
  - GUI(graphic user interface) support
    - LCD
  - Multimedia support
    - Video, Audio
  - Widespread Interface support
    - UART, SPI, USB, ZigBee, Bluetooth

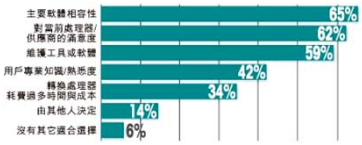
Embedded Microporcessor



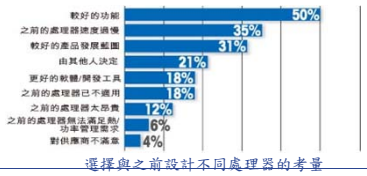
處理器基本調查



主處理器當前設計



選擇與之前設計相同處理器的考量



選擇與之前設計不同處理器的考量

Core Architecture

- 桌面個人電腦主要使用X86 或PowerPC 的中央處理器 (Central Processing Unit, CPU) 核心架構 (core architecture)
- 常見的嵌入式系統 CPU 核心架構有
  - MIPS
  - ARM
  - X86
  - PowerPC
  - SuperH
- 爲減少成本或要符合嵌入式系統的某些特別要求，一些嵌入式系統甚至會利用有特殊用途的積體(集成)電路 (integrated circuit, IC) 來代替一般用途的 CPU

Core Architecture-ARM

- ARM is a family of RISC architectures
- Architectural features of embedded processor
- General rules (with exceptions):
  1. Designed for efficiency (vs. ease of programming)
  2. Huge variety of processors (resulting from 1.)
  4. Heterogeneous register sets
  5. Limited instruction-level parallelism or VLIW ISA
  6. Different operation modes (saturating arithmetic, fixed point)
  7. Specialised microcontroller & DSP instructions (bit-field addressing, multiply/accumulate, bit-reversal, modulo addressing)
  8. Multiple memory banks
  9. No “fat” (MMU, caches, memory protection, target buffers, complex pipeline logic, ...)
- ARM does not manufacture its own VLSI devices
  - Licenses
- ARM7 – Von Neuman Architecture
- ARM9 – Harvard Architecture

Core Architecture-ARM

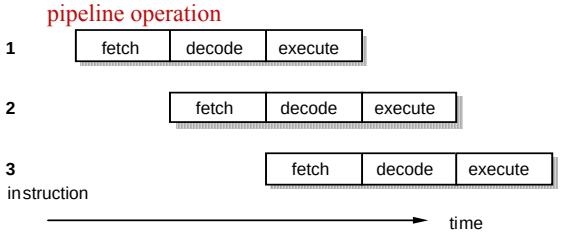
- **Global Partner Network provides a complete system and processor design**

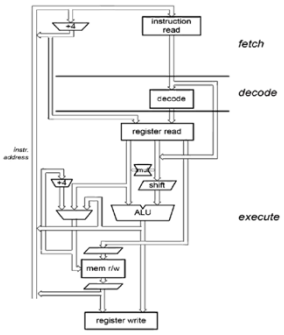
Core Architecture-ARM

- **ARM優點：**
  - 32位元RISC處理器可達到上網或複雜控制的要求
  - 32位元處理器技術的進步及價格下降
  - 高性能、低功率的32位元RISC處理器
  - Thumb 16位元的精簡指令集
  - 三大特點
    - 小體積、抵功率，低成本，高效能
    - 16/32位元指令集
    - 全球最多的合作夥伴
  - 多種擴充系統
    - Thumb
    - DSP：利用DSP做算數指令集
    - Jazeller:允許執行JAVA op code

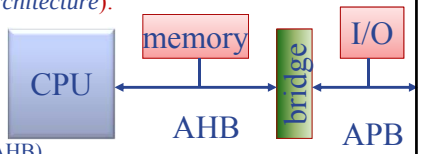
Core Architecture-ARM

- **Example: ARM7TDMI**
  - ARM single-cycle instruction 3-stage pipeline operation





  - AMBA (*Advanced Microcontroller Bus Architecture*):
    - Open standard
    - Many external devices
  - Two varieties:
    - AMBA Advanced High-performance Bus (AHB)
    - AMBA Advanced Peripherals Bus (APB)

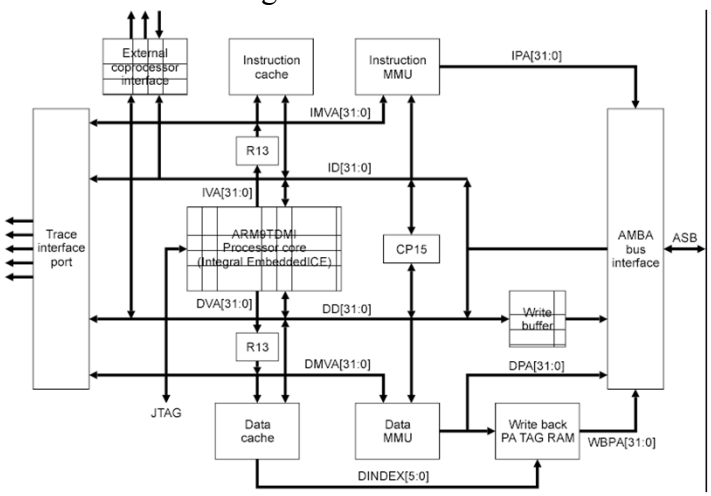


Core Architecture-ARM

- **Example: ARM920T**
  - The ARM920T processor is a member of the ARM9TDMI family of general-purpose microprocessors, includes
    - ARM9TDMI (core)
    - ARM940T (core plus cache and protection unit)
    - ARM920T (core plus cache and MMU)
  - Five-stage pipeline
    - Fetch, Decode, Execute, Memory, and Write
  - Interfaces
    - AMBA *Advanced System Bus* (ASB)
    - AMBA *Advanced High-performance Bus* (AHB)

Core Architecture-ARM

- **ARM920T** - Block Diagram



Core Architecture-ARM

- **ARM920T** - MMU features
  - standard ARMv4 MMU mapping sizes, domains, and access protection scheme
  - mapping sizes are 1MB (sections), 64KB (large pages), 4KB (small pages), and 1KB (tiny pages)
  - access permissions for sections
  - access permissions for large pages and small pages can be specified separately for each quarter of the page (these quarters are called subpages)
  - 16 domains implemented in hardware
  - 64 entry instruction TLB and 64 entry data TLB
  - hardware page table walks
  - round-robin replacement algorithm (also called cyclic)
  - invalidate whole TLB, using CP15 register 8
  - invalidate TLB entry, selected by MVA, using CP15 register 8
  - independent lockdown of instruction TLB and data TLB, using CP15 register 10

嵌入式系統開發板實例

- 華亨科技http://www.hhnet.net.com.tw
- XScale技術的Intel PXA270處理器 520MHZ工作頻率
- 64MB SDRAM
- 64MB Nand Flash
- 16MB Nor Flash
- 8”640*480TFT全彩LCD
- 觸控螢幕
- 可執行Linux 2.6.x及WinCE.net 5.0作業系統

嵌入式系統開發板實例

嵌入式系統開發板實例

XScale

- The **XScale**, a microprocessor core, is Marvell's (formerly Intel's) implementation of the fifth generation of the ARM architecture
- Consists of several distinct families: IXP, IXC, IOP, PXA and CE
- The XScale architecture is based on the ARMv5TE ISA without the floating point instructions
- XScale uses a seven-stage integer and an eight-stage memory superpipelined RISC architecture
- All the generations of XScale are 32-bit ARMv5TE processors manufactured with a 0.18-µm or 0.13-µm (as in IXP43x parts) process and have a 32-KB data cache and a 32-KB instruction cache

Outline

- Introduction to Embedded Systems
- Understanding the ARM serious
- **Famous Embedded OS**
- Linux Overview
- Introduction to Linux Programming
- Guide to Programming on ARM serious
- Embedded Systems Development

Outline

- TinyOS
- Win CE
- Embedded Linux
- Android
- Palm OS
- Symbian
- iOS

TinyOS(1)

- TinyOS是一個開放式的嵌入式作業系統，它是由加州大學的伯利克分校開發出來的，主要應用於無線感測器網路方面。它是基於一種組件（Component－Based）的架構模式，使得能夠快速實現各種應用。
- TinyOS的程式採用的是模組化設計，所以它的程式核心往往都很小（一般來說核心程式和所需資料大概在400 Bytes左右），能夠突破感測器存儲資源少的限制，這能夠讓TinyOS很有效的運作在無線感測器網路上並去執行相應的管理工作等。

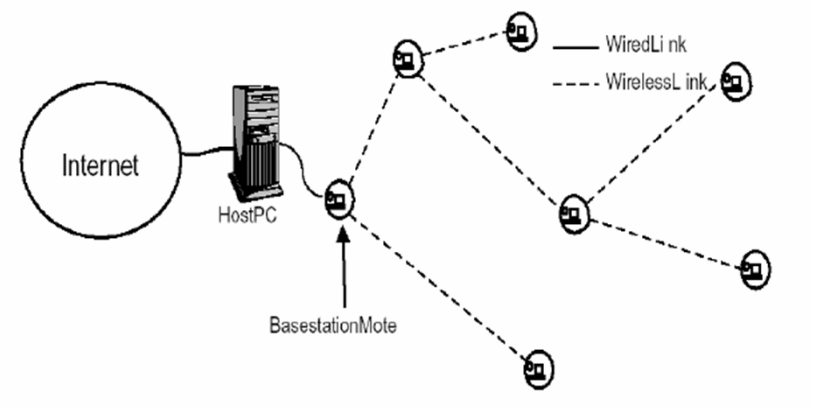
TinyOS (2)

- TinyOS本身提供了一系列的元件，可以很簡單方便的編撰程式，用來讀取和處理感測器的數據並透過無線電來傳輸訊息。
- 可以把TinyOS看成是一個可以與感測器進行交互的API界面，它們之間可以進行各種通訊。

TinyOS (3)

- TinyOS在構建無線感測器網路時，它會有一個基地控制台，主要是用來控制各個感測器子節點，並聚集和處理它們所採集到的訊息。
  TinyOS只要在控制台發出管理訊息，然後由各個節點透過無線網路互相傳遞，最後達到協同一致的目的，比較方便。

TinyOS (4)



TinyOS (5)

- TinyOS的特點
  - Componented-Based Architecture
    - TinyOS提供一系列可重用的元件，一個應用程式可以透過連接配置檔案（A Wiring Specification）將各種元件連接起來，以完成它所需要的功能。
  - Event-Driven Architecture
    - TinyOS的應用程式都是基於事件驅動模式的，採用事件觸發去喚醒感測器工作。

TinyOS (6)

- TinyOS的特點
  - Tasks And Events Concurrency Model
    - Tasks一般用在對於時間要求不是很高的應用中，且tasks之間是平等的，即在執行時是按順序先後來得，而不能互相搶先執行，一般為了減少tasks的營運時間，要求每一個task都很短小，能夠使系統的負擔較輕
    - Events一般用在對於時間的要求很嚴格的應用中，而且它可以搶先優於tasks和其他events執行，它可以被一個操作的完成或是來自外部環境的事件觸發，在TinyOS中一般由硬體中斷處理來驅動事件。

TinyOS (7)

- TinyOS的特點
  - Split-Phase Operations
    - 在TinyOS中由於tasks 之間不能互相佔先執行，所以TinyOS沒有提供任何阻塞操作，為了讓一個耗時較長的操作盡快完成，一般來說都是將對這個操作的需求和這個操作的完成分開來執行，以便獲得較高的執行效率。

TinyOS (8)

- TinyOS的編程
  - nesC—是一種類似C的語言，它是對C的擴展，也是架構化的語言。它是基於元件式的編程，模組化的設計。
  - nesC 元件有兩種：
    - Module（模組）
    - Configuration（連接建置檔案）

TinyOS (9)

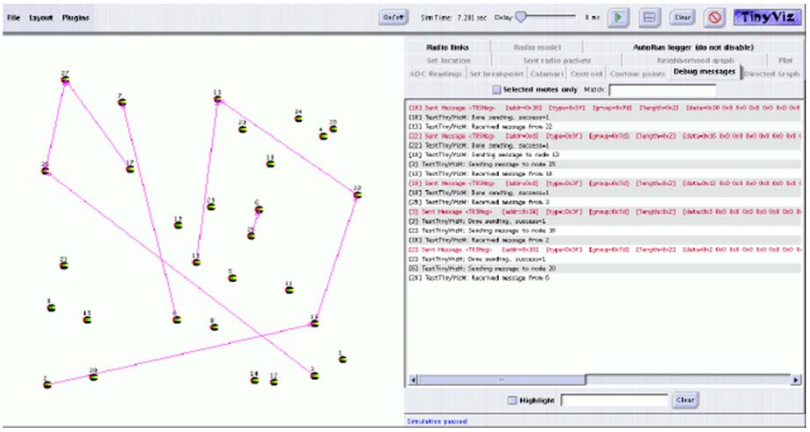
- TinyOS的編程
  - nesC Component
    - Module—在模組中主要實現程式的編製，它可以使用和提供界面，在它的實現部分必須對提供界面裡的command和使用界面裡的event進行實現。
    - Configuration—在連接建置檔案中主要是將各個元件和模組連接起來成為一個整體，它也可以提供和使用界面。

TinyOS (10)

- TinyOS Simulator
  - TOSSIM—它可以同時模擬多個sensor nodes執行同一個程式，提供執行時的調試和配置，它也可以Real Time監測網路狀況，並向網路注入調試訊息、Radio or UART packets等，還可以與網路進行交互。

TinyOS (11)

- TinyOS Simulator

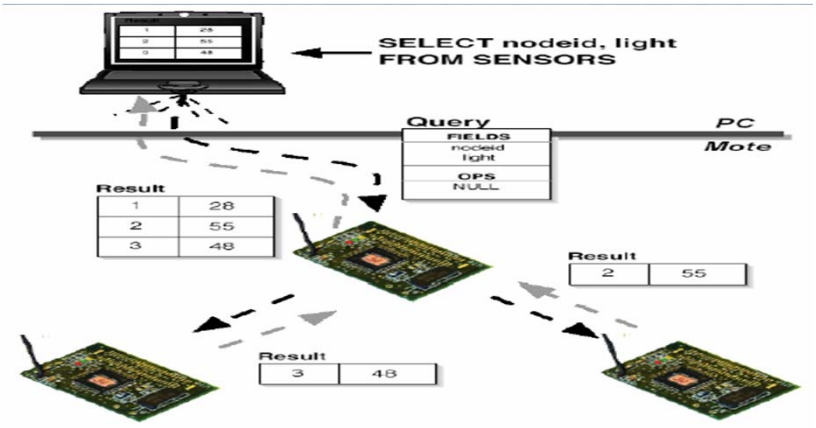


TinyOS (12)

- TinyDB
  - TinyDB 是TinyOS的查詢處理系統，它能夠從無線網路中的sensor節點上提取數據和訊息。
  - TinyDB提供了一個簡單的類似SQL的界面，只要你指定你所感興趣的數據，它就能將它提取出來，並且透過設定適當的參數，還能夠對數據進行過濾和聚集。
  - TinyOS為TinyDB提供了一個可視化的JAVA API視窗，可以進行實時查詢。

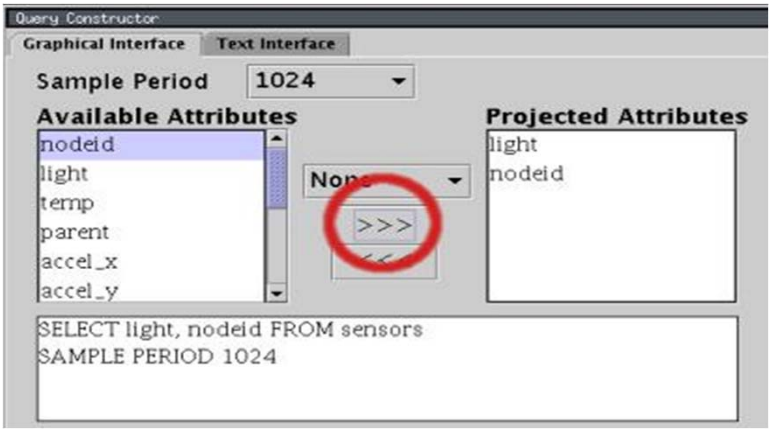
TinyOS (13)

- TinyDB



TinyOS (14)

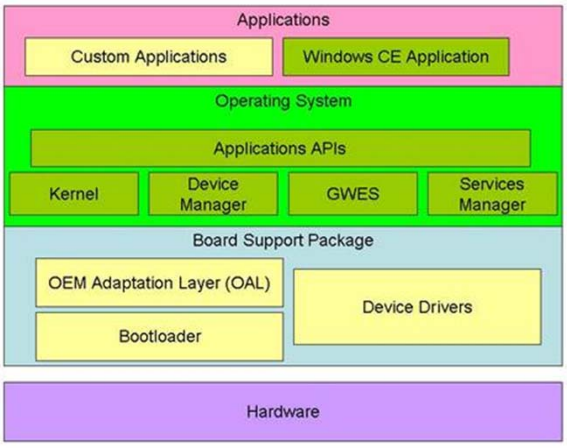
- TinyDB GUI



WinCE(1)

- Windows CE 是微軟專為嵌入式系統所推出的embedded OS，Windows CE 可應用於網路設備、Set-top-Box、工業控制器等嵌入式裝置。
- Windows Mobile 5.0採用Windows CE 5.0的核心，是專為PocketPC與Smartphone所推出的產品。
- WinCE的開發環境可以相當低的花費取得，甚致可以取得免費版本
- Windows Mobile 的開發環境則是必須經由微軟授權後取得。

WinCE(2)



WinCE(3)

- WinCE 的產品開發門檻低、程式開發工具友善使用性佳、使用者介面豐富與完整的多媒體支援，使得WinCE在行動手持裝置的產品開發上具備相當的優勢。
- 除了使用性佳的操作介面與強大的多媒體撥放支援外，更可以執行Mobile Office套裝軟體（Word Mobile、PowerPointMobile Viewer等）、使用MSN與收發Email等。

WinCE(4)

- Windows Mobile 上能執行許多PC上常用的許多應用程式，例如：Mobile IE、Windows Media Player 10 Mobile 5，都是讓Windows Mobile智慧型手機更聰明好用的關鍵。
- 在嵌入式軟體開發方面，Windows Mobile 支援 .NET CF （Compact Framework）
- 開發工具目前也整合成一套Visual Studio 2005，開發Windows Mobile的應用程式更加輕鬆愉快；需要資料庫應用的應用程式，也能使用SQL Mobile 解決方案。

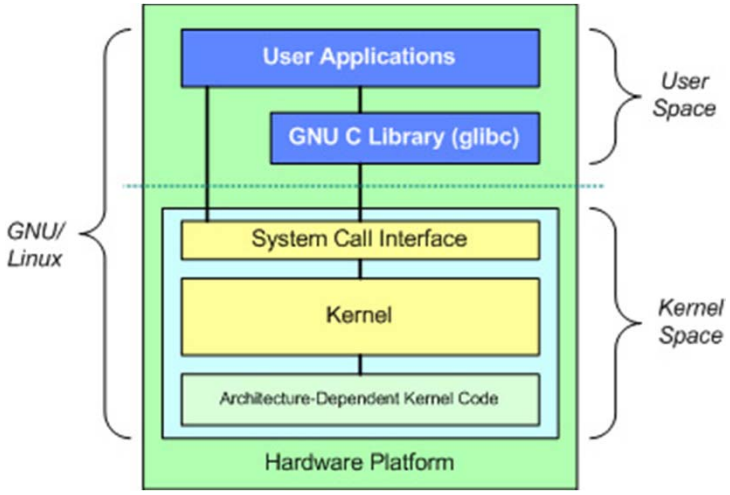
Embedded Linux(1)

- Embedded Linux其實並不是一個作業系統，而是代表「應用Linux系統於embedded system」的名詞。
- 。Embedded Linux的技術核心主軸是在研究「如何將Linux系統嵌入至嵌入式目標裝置裡」。
- 嵌入式Linux系統的技術主軸有二：(1) Linux kernel 與 (2)Root file system
  - Root file system即是「小型的Linux系統」，root file system裡頭存放Linux的系統架構（file system hierarchy）、指令、工具、應用程式、shared libraries、驅動程式等。

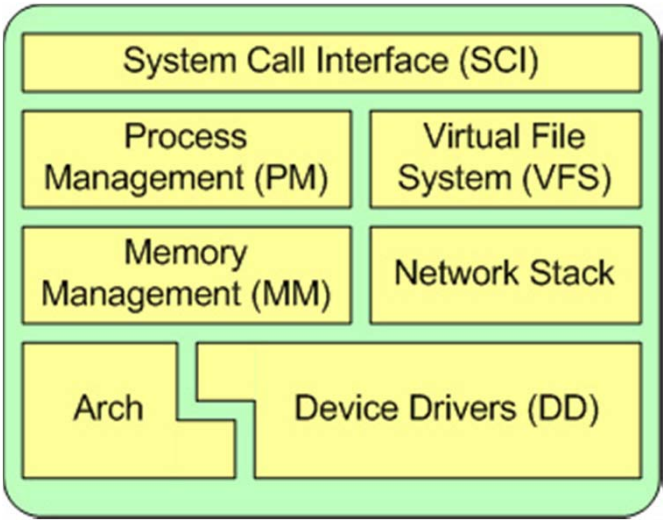
Embedded Linux(2)

- 相較於WinCE 友善的程式開發環境，Embedded Linux的開發環境則是比較不容易上手的。
- 目前專門針對Embedded Linux的開發工具尚不成熟，因此我們都是在Linux的”PC”上使用Linux一般性的程式開發工具來進行嵌入式Linux的發展。
- 但是，相對於WinCE的架構複雜性，Linux則是提供簡易的程式開發。

Embedded Linux(3)



Embedded Linux(4)



Embedded Linux(5)

- Embedded Linux的開發工具，主要以GCC和glibc為核心。
  - GCC（http://www.gnu.org/software/gcc/gcc.html）是GNU Compiler Collection的縮寫，也就是許多編譯器的收集，目前支援的程式語言有：C、C++、Objective-C、Fortran 、Java、Ada。GLIBC是GNU的C標準程式庫，
  - GLIBC提供system call的界面函數與標準的C 函數。GLIBC也符合許多標準與規格，讓使用GLIBC的程式可以更容易移植到其它UNIX平臺。

Android(1)

- **Android**是基於**Linux內核**的**軟體平台**和**作業系統**
- 是Google在2007年11月5日公佈行動**電話**系統平台
- 早期由Google開發，後交由開放手機聯盟（Open Handset Alliance）持續開發

Android(2)

- 採用了軟體堆層（software stack，又名以軟體疊層）的架構，主要分為三部分
  - Application Framework
  - Library and Virtual Machine
  - Linux Kernal
- 最底層以**Linux內核**工作為基礎，只提供基本功能；其他的應用軟體則由各公司自行開發，以Java作為編寫程式的一部分

Android(3)

Android(4)

Android(5)

- 目前Android 的 Linux kernel控制包括安全(Security)，記憶體管理(Memory Management)，程式管理(Process Management)，網路堆疊(Network Stack)，驅動程式模型(Driver Model)等
- 下載Android源碼之前，先要安裝其構建工具 Repo來初始化源碼。Repo 是 Android 用來輔助Git工作的一個工具。
- Android的版本目前發佈到3.5，但是有明確功能描述者為2.2與3.0版

Palm OS(1)

- **Palm OS**是早期由**U.S. Robotics**（其後被3Com收購，再獨立改名為Palm公司），研製專用於其**掌上電腦產**品Palm的操作系统
- 由於此操作系统完全為Palm產品設計和研發，而其產品由推出時就超過了Apple公司的**Newton**而獲得了極大的成功，所以Palm OS也因此聲名大噪。其後曾被IBM、Sony、Handspring等廠商取得授權，使用在旗下產品

Palm OS(2)

- Palm OS操作系统以簡單易用為大前提，運作需求的記憶體與處理器資源較小，速度也很快；但不支援多執行緒(Multi-Threads)
- Palm OS版權現時由PalmSource公司擁有，並由PalmSource開發及維護。2005年9月9日，PalmSource被日本軟件開發商愛可信收購
- Palm OS 目前被使用於產品的版本到5.0

Symbian(1)

- **Symbian作業系統**是**Symbian公司**為**手機**而設計的**作業系統**，被Nokia收購之後，將其移轉到Symbian基金會，以開放原始碼的形式釋出
- 它的前身是Psion的EPOC，並且獨佔式的執行於**ARM**處理器。包含由**Symbian公司**所提供的相關的函式庫（libraries）、使用者介面（user interface）架構和共用工具（common tools）的參考實作（reference implementation）

Symbian(2)

- Symbian是以EPOC為基礎，而它的架構於許多桌上型作業系統相似，它包含**先佔式多工**、多**執行緒**和記憶體保護等
- Symbian的最大優勢是在於它是為便攜式裝置而設計，而在有限的資源下，可以執行數月甚至數年

Symbian(3)

- 而這要歸功於節省記憶體，使用Symbian風格的編程理念是清除堆疊。
- 將這些功能與其他技術搭配使用，會使記憶體使用量降低且記憶體泄漏量極少

- 類似技術也運用於節省磁碟(儘管在Symbian裝置中，硬碟通常指快閃記憶體)和記憶卡使用空間

Symbian(4)

- Symbian的編程是使用事件驅動，當應用程式沒有處理事件時，CPU會被關閉

- 這是透過一種叫主動式物件的編程理念進行實作的，正確的使用這些技術將能夠延長電池使用時間。 這些技術讓Symbian的C++變得非常專業。然而，許多Symbian的裝置也可以利用OPL、Python、Visual Basic、Simkin以及Perl來搭配J2ME和自行開發的Java來使用

Symbian(5)

- Symbian有個微核心架構，這定義了核心內部所必需的最少功能。微核心架構包含排程系統和記憶體管理，但不包含網路和檔案系統支援
- 基本層則是再由
  - 檔案伺服器
  - 網路及通訊子系統
  - 使用者介面碼

所組成

- Symbian目前已開發至9.0版

iOS(1)

- **iOS**是由Apple公司為iPhone開發的操作系統，它主要是給iPhone、iPod touch以及iPad使用
- iOS源自蘋果原有的麥金塔個人電腦(Macintosh；Mac)作業系統Mac OS X，兩者均以Darwin為基礎所發展成

iOS(2)

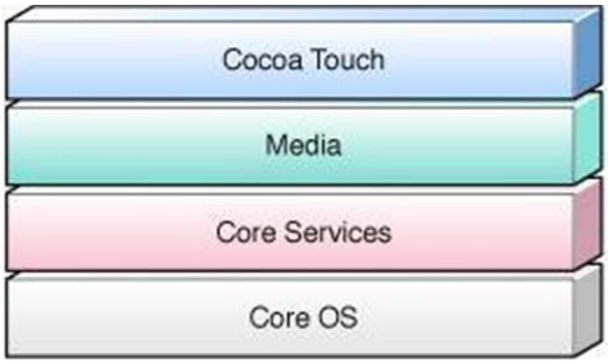
- Darwin，是一個工業級、基於UNIX的平臺，它的目標是為了得到出眾的穩定性和性能
- Darwin是蘋果工程師和開放源代碼軟體共同體的編程人員的共同結晶
- 在Darwin的核心使用的是BSD

iOS(3)

- iOS的系統架構分為四個層次：
  - 核心作業系統層（the Core OS layer）
  - 核心服務層（the Core Services layer）
  - 媒體層（the Media layer）
  - 可輕觸層（the Cocoa Touch layer）

iOS(4)

- iOS的系統架構分為四個層次：



iOS(5)

- iOS更進階的技術都是以C以及Objective-C 做為基礎的介面
- iPhone和iPod Touch使用基於ARM架構的中央處理器，因此，Mac OS X 上的應用程式不能直接複製到iOS上執行。他們需要針對iOS的ARM重新編寫

Outline

- Introduction to Embedded Systems
- Understanding the ARM serious
- Famous Embedded OS
- **Linux Overview**
- Introduction to Linux Programming
- Guide to Programming on ARM serious
- Embedded Systems Development

Linux Overview

- 由芬蘭 Linuz Torvalds 於1991. 10. 創始
- 承襲開放式原始碼(Open Source)的精神
- 經由許多UNIX的程式設計師與Internet程式設計師集合創作與演化
- 一套免費且功能完整的UNIX軟體、是一個32位元的作業系統，運作穩定且有效率
- 是一套專為為80x86個人電腦所設計的UNIX作業系統
- Linux核心不使用AT&T或其他專屬性的程式碼；主要軟體來至MIT免費軟體的GNU專案

Linux Overview

- 依循POSIX標準，與UNIX完全相容
- 支援完整的網路軟體，包括TCP/IP、Email、Emacs、X Window、NFS、NIS、News、WWW等等
- 提供完整的說明文件，及免費的系統原始程式
- 在安裝與管理上擺脫了UNIX一般給人望之卻步的感覺
- 相關網址
  - http://www.linux.com/
  - http://www.geocities.com/d64332/
  - http://www.redhat.com/
  - http://www.slackware.com/

Linux Overview

- Linux Distribution (Linux 家族)
  - RedHat
  - Slackware
  - Fedora Core
  - Mandriva
  - Debian
  - Gentoo
  - 超過20種以上 ，使用的系統核心，其實都是相同的，『標準』操作的方式(文字模式)也幾乎大同小異

Linux Operations

- 常用指令
- 編輯程式
- 檔案管理

Linux 常用指令

- UNIX指令格式

  指令(Command) 選項s(Options) 參數s(arguments)

  - UNIX指令三大部分之間至少須有一個空白鍵(Space) or <TAB>鍵區隔之選項一律以 - 為前導
  - 三大部分之順序不可更動
  - UNIX系統中，指令、選項及參數均為Case Sensitive，亦即大小寫不可任意更換
  - 同一行中可輸入一個以上之指令，但指令之間必須以分號”;”分隔開
  - 常見命令提示符號 **$**

Linux 常用指令

- **alias**
  - 功能：指定指令的別名
  - 語法：alias "別名"="指令名稱"
  - Example:

    $alias del="rm -i"

- **bg**
  - 功能：將指定的工作(job)移至背景執行
  - 語法：bg "工作代號 "
  - Example:

    $jobs (可顯示目前有幾項Jobs在執行中)

    $bg %1 (把第一項Job移至背景執行)

Linux 常用指令

  - 補充說明：
    - Linux中一個login Session只能有一個前景程式存在，因此讓某一程式進入背景執行為兩以上的程式同時執行的方法
    - 另一個讓程式進入背景的方法為:執行程式時，在程式名稱之後加一個"&"符號。例如："find / -name inetd.conf &"

- **cat**
  - 功能：印出檔案內容至銀幕（標準輸出）或合併多檔。cat file1 file2 > file3可將檔案file1、file2之內容依順序合併，並將結果存至file3
  - 語法：cat [-選項] file
  - Example:

    $cat myfile

Linux 常用指令

- **cd**
  - 功能：改變目前工作目錄之位置至directory，若[directory]略之，則返回至使用者之簽入時之目錄(Home Directory)
  - 語法：cd [目的目錄名稱]
  - Example:

    $cd /bin

- **clear**
  - 功能：清除目前銀幕顯示
  - 語法：clear
  - Example:

    $clear

Linux 常用指令

- **cp**
  - 功能：檔案複製。若target為檔案名稱，則file只能指定一個檔案，若target為一個目錄名稱，則會將file1、file2..分別複製至此一目錄之下
  - 語法：cp[選項] file1 file2或cp [選項]file directory
  - Example:

    $cp -i test.tar warnning

    選項"i"若target存在，會要求使用者確認複製之動作

Linux 常用指令

- **df**
  - 功能：顯示磁碟相關資訊
  - 語法：df [選項]
  - Example:

    $df
  - 可用選項及代表意義:

    -a 所有檔案系統(Partition)
    -i 顯示inode資訊
    -k 以KByte為單位顯示磁碟資訊
    -l 僅顯示Local File Syatem(不含NFS)
    --help 顯示獻上求助資訊(關於df)

Linux 常用指令

- **exit**
  - 功能：退出系統(=logout)
  - 語法：exit
  - Example:

    $exit
- **fg**
  - 功能：將指定的工作(job)移至前景景執行
  - 語法：用法同 "bg“
  - Example

    $fg
  - 當你把程式移至前景，你將無法繼續在終端機上輸入指令

Linux 常用指令

- **find**
  - 功能：依照指定條件，找尋檔案或目錄
  - 語法：find start_directory [-選項]
  - Example:

    $find ./ -size +500 –print

    找到目前目錄下，檔案大小大於250K Bytes的檔案，並將結果由終端機輸出
  - 常用選項及代表檔案條件

    -amin <min> 指定時間內曾被存取過的檔案，單位為分鐘

    -atime <day> 指定時間內曾被存取過的檔案，單位為24小時

    -exec <指令> 當找到符合條件的檔案，就執行該指令，指令結尾必須是";“

    -mmin <min> 指定時間內曾異動過的檔案，單位為分鐘

    -mtime <day> 指定時間內曾被異動過的檔案，單位為24小時

    -nouser 檔案不屬於本機上之任何User的檔案

    -ok <指令> 用法同-exec，唯每一個指令執行前均會詢問使用者

    -print 將符合條件的檔案資料由終端機顯示出來

    -size <檔案大小> 符合檔案大小的檔案，單位為512Byte(Block)，檔案大小前加"+"號代表大於這個 size，"-"代表小於這個size

    -user <username> 屬於"username"的檔案

Linux 常用指令

- **grep**
  - 功能：找尋並印出files中含有字串pattern 所在行內容
  - 語法：grep pattern file_name
  - Example:

    $grep woody /etc/passwd

    找出/etc/passwd中含有woody字串的行

    $ps aux | grep woody

    找出目前系統中，所有以woody名義執行的process訊息

Linux 常用指令

- **gunzip**
  - 功能：解開壓縮檔(.gz)
  - 語法：gunzip file_name
  - Example:

    $gunzip a.gz

    解開檔名為a.gz的壓縮檔，並刪除a.gz

    $gunzip a.gz > b

    解開檔名為a.gz的壓縮檔，保留a.gz，並將結果輸出至"b"

Linux 常用指令

- **gzip**
  - 功能：檔案壓縮
  - 語法：gzip file_name
  - Example:

    $gzip a.out
  - **gzip -d = gunzip**
- **last**
  - 功能：列出目前與過去登入系統的使用者相關資訊
  - 語法：last [帳號名稱]
  - Example

    $last alex
  - 列出alex使用者的登錄資訊

Linux 常用指令

- **ls**
  - 功能：列出所在或指定目錄的內容or檔案之相關特性
  - 語法：ls [-選項] [目錄or檔案名稱]
  - Example:

    $ls -l
  - **常用選項及其代表之意義**

    -a : 列出所有檔案名稱，包括以『.』開頭之隱藏檔，如.profile、.login等

    -d : 若參數的為一目錄名稱，則只列出檔案名稱而非目錄之內容。

    -l : 列出檔案的詳細資料，包括檔案形態、存取權限、連結數目、擁有 者名稱、群組名稱等。

Linux 常用指令

- **mail**
  - 功能：E-Mail收送管理程式
  - 語法：mail [-b <bcc_list>] [-f <mail_file>] [-s <subject>] [-u <user>] [mail Address]
    Example:

    $mail -v -s "Mail Test" alex@gmail.com
  - 寄一封信給woody,信件主旨為“Mail Test”，並於enter之後開始繕打信件內容，直到輸入quit離開

Linux 常用指令

- **man**
  - 功能：線上求助程式
  - 語法：man <指令名稱>
    Example:
    $man ls
  - 查詢ls指令的語法
- **mkdir**
  - 功能：建立目錄
  - 語法：mkdir <目錄名稱>
  - Example:
    $mkdir movie

Linux 常用指令

- **more**
  - 功能：以每次一頁方式顯示一個檔案
  - 語法： more file_name
  - Example
    $more myfile.txt
- **passwd**
  - 功能：更改使用者密碼
  - 語法：passwd [user_name]
  - 補充說明 :
    1.當user_name不提供時，為更改目前login者密碼
    2.更改本身密碼時，必須先輸入舊密碼，確認無誤後，才可更改
    3.不具備MD5功能的系統，密碼最大長度為8字元

Linux 常用指令

- **mv**
  - 功能：搬移檔案 /目錄
  - 語法： mv 檔案名(目錄名) 目的目錄名
  - Example

    $mv filename targetdirectory

- **ps**
  - 功能：列出所有的工作
  - 語法：ps
  - Example

    $ps

Linux 常用指令

- **w (who)**
  - 功能：列出目前線上使用者資訊
  - 語法： w (或是 who)
  - Example

    $who

- **whereis**
  - 功能：在指定或特定的目錄中尋找檔案
  - 語法：whereis file_name
  - Example

    $whereis myfile.c

Linux 常用指令

- **rmdir**
  - 功能：刪除空目錄
  - 語法： rmdir 空目錄名 ，必須是空目錄
  - Example

    $rmdir olddirect

- **rm**
  - 功能：刪除檔案
  - 語法：rm file_name
  - Example

    $rm myfile.c

Linux 常用指令

- **kill**
  - 功能：刪除執行程式(行程)
  - 語法： kill process_id
  - Example

    $kill 2384

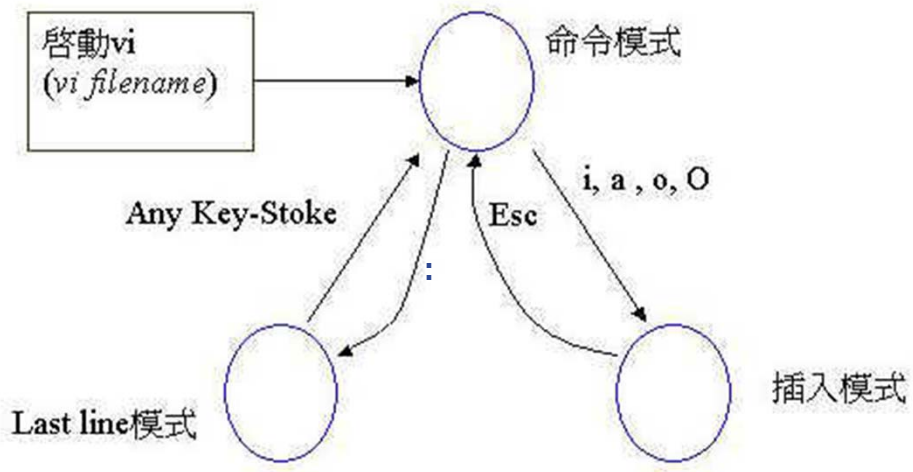
- **pwd**
  - 功能：顯示目前的工作目錄
  - 語法：pwd
  - Example

    $pwd

編輯軟體—vi 及vim

- vi是全銀幕文字編輯器
- 不是僅有，也不是最好用，卻是唯一存在所有UNIX版本中之Editor
- 三種模式：命令模式、插入（輸入）模式、last line(延伸命令)模式

編輯軟體—vi 及vim



編輯軟體—vi 及vim

- vi啟動
  - $ vi filename
- 命令模式
  - 命令模式基本指令可分為游標移動、插入文字、刪除文字、修改文字等幾類
  - 熟悉各種常用指令，並善加組合變化，可使文字編輯的工作變得快速而有效率

編輯軟體—vi 及vim

- last line模式
  - last line模式之指令一般用於輔助命令模式指令之不足
  - 我們可以將last line模式指令視為 ":"開頭的命令模式指令
  - 主要指令如下
    - :q 不儲存檔案內容並離開vi(未變更檔案內容)
    - :q! 不儲存檔案內容並離開vi(檔案內容已被更動)
    - :wq <filename> 儲存檔案內容並離開vi
    - :w <filename> 儲存檔案
    - := 顯示游標目前所在之行數
    - :.= 顯示檔案全部總行數
    - :n,ms/str1/str2/opt從第n行搜尋至第m行，並將所找到之字串" str1"取代為" str2"。opt=g全部取代，opt=c確認再取代
    - :r filename將外部檔案引用(paste)至游標所在位置之後
    - :e filename編輯另一檔案
    - :! Command 執行SHELL command後返回

編輯軟體—vi 及vim

- 命令模式指令概述
  - 游標移動指令
    - k 往上移一格
    - j 往下移一格
    - h 往左移一格
    - l 往右移一格
    - w往下移一個字(word)
    - b 往上移一個字(word)
    - 1 or 0 移至游標所在行首
    - $ 移至游標所在行首
    - n G 移至第n行
    - G 移至檔尾

編輯軟體—vi 及vim

- 命令模式指令概述
  - 插入本文
    - a 在游標後插入文字（進入插入模式）
    - A 在行尾插入文字（進入插入模式）
    - i 在游標前插入文字（進入插入模式）
    - I 在本行前插入文字（進入插入模式）
    - o 在本行之下開新一行並輸入
    - O 在本行之上開新一行並輸入
    - <Ctrl> v 輸入特殊字元(在插入模式下)

編輯軟體—vi 及vim

- 命令模式指令概述
  - 編輯指令
    - nyy copy n行資料放置緩衝區中
    - y+ 游標移動指令 copy游標移動範圍之資料至緩衝區中 (e.g.:y3w: copy游標之後三個字)
    - p 將緩衝區之資料copy至目前游標所在
    - nx 刪除游標之後n個字元
    - ndd 往下刪n行
    - d+ 游標移動指令 copy刪除游標移動範圍之資料 (e.g.:dw刪一個字)

編輯軟體—vi 及vim

- 命令模式指令概述
  - 其他
    - /字串<CR> 尋找字串所在位置（往下）
    - ?字串<CR> 尋找字串所在位置（往上）
    - u 放棄上一個指令動作
    - . 重複上一個本文更改指令

Linux 檔案系統

- 檔案系統
  - 實體上而言(Physically)，UNIX中所謂”檔案系統”，相當於DOS中的partition
  - 一個獨立的檔案系統擁有獨立的file table(inode table)
  - 一個磁碟中，可以切割成一個上的檔案系統
  - 但在邏輯上而言(Logically)，一個UNIX中，僅有一個檔案系統
  - 亦即，一個UNIX作業系統中僅有一個“根目錄“，一切檔案結構依此根目錄，往下發展

Linux 檔案系統

- 檔案系統架構
  - UNIX檔案系統和DOS一樣，採階層式(Hierarchy)或樹狀(Tree)之目錄架構
  - 最上一層目錄稱為根目錄(root)
  - 每一目錄中均可含有檔案與子目錄，路徑名稱中目錄與目錄之分隔符號為”/”。

Linux 檔案系統

- 檔案種類
  - 一般檔案(ordinary files)，如原始程式、文件、資料…等
  - 目錄檔案(directory files)，目錄檔用於涵括檔案，使檔案系統更加簡潔
  - 特殊檔案(Special device files)，代表某種特殊硬體設備，如：印表機、磁碟機、磁帶機…等，通常存在於/dev目錄之下。
    - 此種檔案類型又可區分為區塊(block)及字元(Character)兩種。(UNIX作業系統下，所有設備均可視為檔案，並依檔案方式運作)
  - 符號鏈結檔(Symbolic link)，內容為一指標，指向檔案名稱所在(非檔案內容儲存所在)；類似檔案別名

Linux 檔案系統

- 檔案命名原則
  - 可用任何字元來命名(“/”除外)，然特殊字元(*?^<…)，除”.”外最好避免使用，免生誤解
  - 早期UNIX檔名長度限制不可超過14個字元，現今UNIX系統則可允許檔名長度至256個字元
  - UNIX檔案系統中，無所謂延伸檔名，對UNIX而言檔名:file.a.b.c.kkk是合法的
  - 以”.”開頭之檔名，為隱藏檔，需用”ls -l“才可看到，如.profile
  - 一個檔案可具有一個以上的檔名，此稱檔案多重鏈結

Linux 檔案系統

- 特殊目錄與路徑名稱
  - 工作目錄(working directory)：目前所在目錄，可以pwd指令顯示之。
  - 家目錄(home directory)：帳號建立之初，管理者賦予使用者之初始目錄；通常這是使用者進入系統之後的第一個工作目錄。”cd”不加任何參數可使使用者回至家目錄。
  - “.” & ”..”: “.”代表目前所在目錄，”..”則代表上一層目錄。
  - 根目錄(root) “/”：整個UNIX檔案系統最上層之目錄。
  - 絕對路徑：以”/”起頭之路徑名稱，如/usr/var/spool/mqueue。UNIX中，每一個目錄或檔案，皆有一個唯一的絕對路徑與之對應。
  - 相對路徑: 起頭不是”/”的路徑，相對路徑是以目前工作目錄為起點的檔案或路徑描述。
    - 如：假設目前工作目錄為/home/center，則相對路徑”work1/papers/lect”與絕對路徑”/home/center/work1/papers/lect”的表示方法，所指的檔案是相同的

Linux 檔案系統

- 檔案相關訊息
  - 檔案模式
    - -: 一般檔案
    - d:目錄
    - b:特殊檔案(區塊)
    - c:特殊檔案(字元)
    - l:符號連結檔

Linux 檔案系統

- 檔案相關訊息
  - 檔案存取權限
    - 使用者區分:檔案擁有者、群組、其他
    - 權限區分：讀(r)、寫(w)、執行(x)
    - UNIX共用九個字元來描述三種使用者之三種權利
- 鏈結個數
  - 指向此一檔案之檔案名稱(inode)之個數，此鏈結個數不包含符號鏈結

Linux 檔案系統

- 檔案擁有者
  - 擁有此一檔案之使用者名稱
- 群組
  - 此一檔案所屬之群組
- 檔案大小
  - 單位為byte

Linux 檔案系統

- 修改日期
  - 此檔案最後一次被修改日期
- 檔名
  - 檔案名稱
- 範例說明

-rw-r--r--1 root root 268 Jun 28 13:05 /etc/lilo.conf

Linux 檔案系統

- 檔案存取保護
  - UNIX檔案系統將使用者分成檔案擁有者、群組及其他三大類；存取權力則分成讀取(r)、寫入(w)及執行(x)三種；藉著使用者區分與存取權力之給予之組合，組成UNIX的檔案保護系統
  - 檔案擁有者
    - 每一個檔均有一個擁有者(owner)，僅有owner及super user可以更改檔案之存取保護模式(with command chmod, 說明容下文敘述)，而一般則只有super user可以更改檔案之擁有者(with command “chown”)

Linux 檔案系統

- 檔案存取保護
  - 檔案存取權限說明
    - 讀取(r)：可讀取檔案內容，複製檔案內容
    - 寫入(w)：可更改檔案內容、名稱（搬移檔案），刪除檔案
    - 執行(x)：可執行檔案。(If it's a executable file，包括shell program)
  - 目錄存取權限說明
    - 讀取(r)：可瀏覽目錄內容(ls directory)
    - 寫入(w):可在目錄中建立或刪除檔案
    - 執行(x)：可搜尋此目錄，並可更改工作目錄(cd)至此目錄
      - cat /var/spool/mqueue/syslog
      - 這個指令的執行，就必須對/var/spool/mqueue這個目錄具有執行之權力

Linux 檔案系統

- 檔案權限修改
  - 只有檔案擁有者及SuperUser可以更改檔案權限
  - 只有SuperUser可以更改檔案擁有者
  - 檔案權限修改指令 -- chmod，其語法如下：
    - chmod {a, u, g, o}{+,-}{r,w,x} filename(相對表示法)
    - chmod x1x2x3 filename(絕對表示法)

      a：所有User
      u：檔案擁有者
      g：群組
      o：其他User
      +：增加權力
      -：減少權力
      r：讀取
      w：寫入

Linux 檔案系統

- 檔案權限修改
  - x1x2x3：x1、x2、x3均為“三個二進位數字所組成之十進位數字表示“（範圍0~7）
  - x1表owner權限，x2表群組user權限，x3則表其他User之權限
  - 例如755 (=111 101 101)表示檔案存取權限為: rwx r-x r-x
    - Owner有rwx的權利
    - User及Other只有r與x的權利
  - 例如 chmod 755 myfile.c

Linux 檔案系統

- 使用範例

$ls -l kerm

-rwxr-xr-x 1 woody users 164 Feb 4 14:49 kerm*

$chmod g+w,o-x kerm (= chmod 774 kerm)

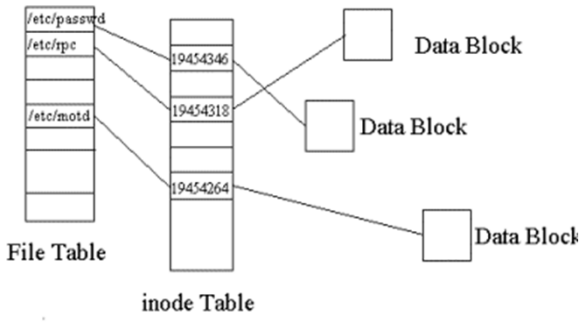
$ls -l kerm

-rwxrwxr-- 1 woody users 164 Feb 4 14:49 kerm*

Linux 檔案系統

- 檔案鏈結(File Link)

UNIX File System Structure



Linux 檔案系統

- 檔案鏈結(File Link)
  - UNIX系統事實上是以所謂的inode號碼來識別檔案
  - 用ls -i，你可以看到每一個檔案的indoe number
  - 利用檔案鏈結可以使一個檔案擁有多個檔名
  - Hard Link
    - 指令：ln file1 file2
      - file1為已存在之檔案，file2則是以上指令所產生之檔案鏈結
      - Hard Link的結果是”在File Table中產生一個新的檔名，指向一個已存在的inode”
      - 地位上file1與file2相同，file1與file2必須同時被刪除才能真正刪除原有之檔案
      - Hard Link不允許跨過file system之鏈結

Linux 檔案系統

- 檔案鏈結(File Link)
  - Symbolic Link
    - 指令：ln -s file1 file2
      - file1為已存在之檔案，file2則是以上指令所產生之符號鏈結
      - Symbolic Link的結果是” 在File Table中產生一個新的檔名file2，指向另一個已存在之檔名file1”
      - 基本上刪除file1，該檔案之內容便已被刪除，與file2會隨之刪除
      - Symbolic Link允許跨過file system之鏈結

Linux 檔案系統

- 使用說明

$ls -l kerm*

-rwxrwxr-- 1 woody users 164 Feb 4 14:49 kerm

$ ln kerm kerm1 (產生一個Hard Link)

$ ls -l kerm* (請注意link數之改變及兩個檔案的file size)

-rwxrwxr-- 2 woody users 164 Feb 4 14:49 kerm

-rwxrwxr-- 2 woody users 164 Feb 4 14:49 kerm1

Linux 檔案系統

- 使用說明

$ln -s kerm kerm2 (產生一個Symbolic Link)

$ls -l kerm* (請注意kerm2之表示方式與file size)

-rwxrwxr-- 2 woody users 164 Feb 4 14:49 kerm

-rwxrwxr-- 2 woody users 164 Feb 4 14:49 kerm1

lrwxrwxrwx 1 woody users 4 Feb 4 16:26 kerm2 ->
kerm

Linux 檔案系統

- 使用說明

$rm kerm (刪除kerm檔)

$cat kerm2 (刪除kerm後kerm2之內容亦隨之消失)

cat: kerm2: No such file or directory

$cat kerm1 (kerm1內容依然存在)

-rwxrwxr-- 2 woody users 164 Feb 4 14:49 kerm1

Outline

- Introduction to Embedded Systems
- Understanding the ARM serious
- Famous Embedded OS
- Linux Overview
- **Introduction to Linux Programming**
- Guide to Programming on ARM serious
- Embedded Systems Development

Introduction to Linux Programming

- 安裝Cygwin、GCC 的編譯環境
- 認識GCC、make 的編譯方式

GNU

- GNU 是一種安裝在Linux 作業系統的開發工具
- 安裝GNU 之前要先準備一個Linux 作業系統
- Cygwin 是一套在Windows 系統上提供UNIX 環境的程式集合
- 在Cygwin 裡中安裝GNU 工具後， 使用者就可以在Cygwin 上開發嵌入式應用程式

Cygwin

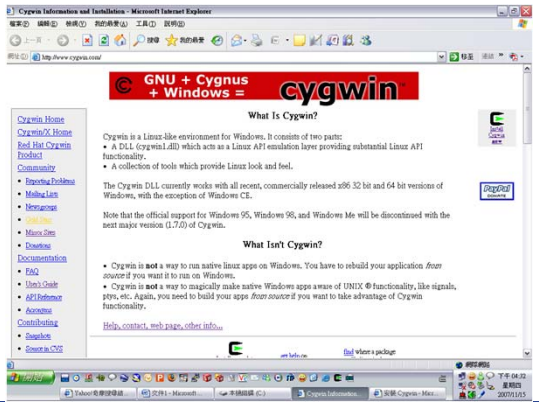
- Cygwin 是一套在Windows 系統上提供UNIX 環境的程式集合，由Red Hat 公司所開發
- 它以Cygwin 資料庫作為應用， 可以產生同於Unix 系統的環境及程式效果
- 使用此一工具， 可以編寫出Win32介面或是GUI 應用程式， 完全符合Win32 API 或是Cygwin API，可以不必改變原先Unix 程式的來源碼就可以直接編譯成Windows 環境用的軟體

Cygwin

- 此程式內建了許多GNU 軟體， 包含Cygwin 開發工具。整套工具包含兩個部份：
  - 一個DLL (cygwin1.dll) 檔案，擔任UNIX 模擬層的角色，提供基本UNIX API 功能
  - 一組移植自UNIX 的工具程式，提供延續自UNIX/Linux 使用習慣的功能

Cygwin

- 從網路下載安裝程式
  - 鏈結網址http://www.cygwin.com/



Cygwin

- Cygwin is a Linux-like environment for Windows.
- It consists of two parts:
  - A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality.
  - A collection of tools which provide Linux look and feel.

Cygwin

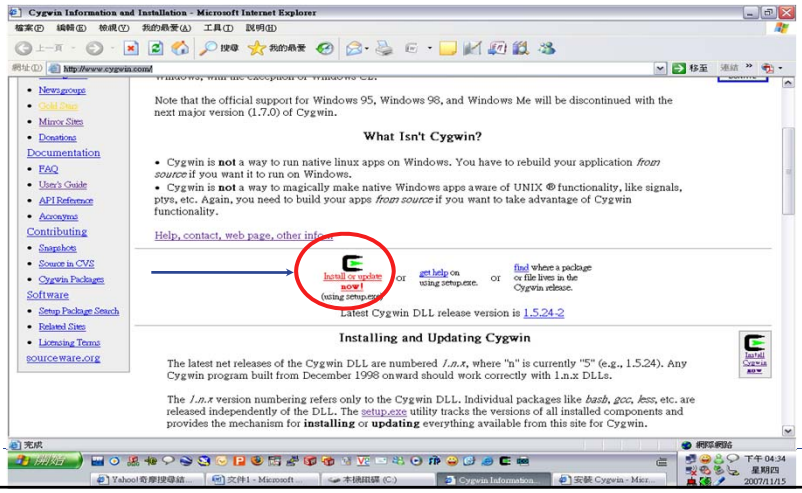
- The Cygwin DLL currently works with all recent, commercially released x86 32 bit and 64 bit versions of Windows, with the exception of Windows CE.
- Note that the official support for Windows 95, Windows 98, and Windows Me will be discontinued with the next major version (1.7.0) of Cygwin.

What Isn't Cygwin

- Cygwin is **not** a way to run native linux apps on Windows. You have to rebuild your application *from source* if you want it to run on Windows.
- Cygwin is **not** a way to magically make native Windows apps aware of UNIX ® functionality, like signals, ptys, etc. Again, you need to build your apps *from source* if you want to take advantage of Cygwin functionality.

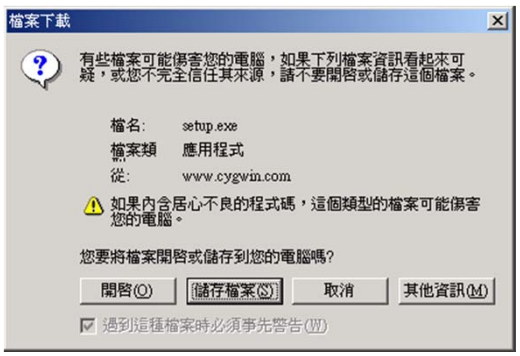
Download Cygwin

- 按箭頭處下載cygwin



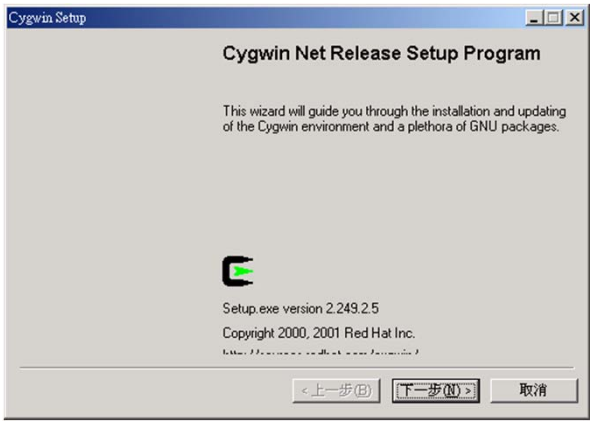
Download Cygwin

- 下載檔案為 setup.exe
- 可直接點選執行安裝程式，或先行下載至個人電腦



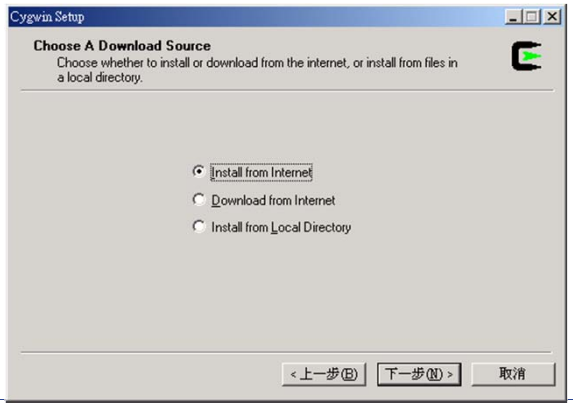
Download Cygwin

- 執行安裝程式後，第一個出現的畫面



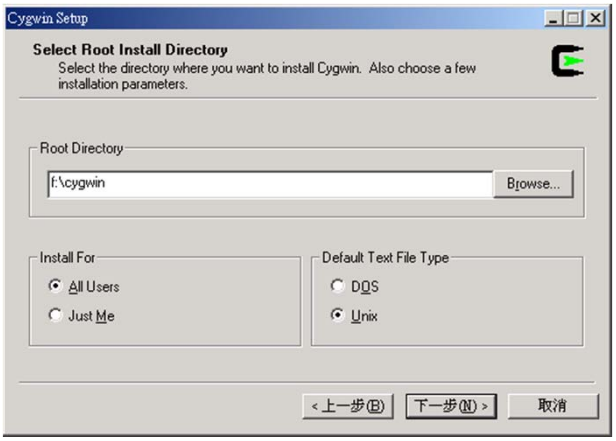
Download Cygwin

- 選擇安裝所需程式來源的方法
  - 有3種方式，如圖



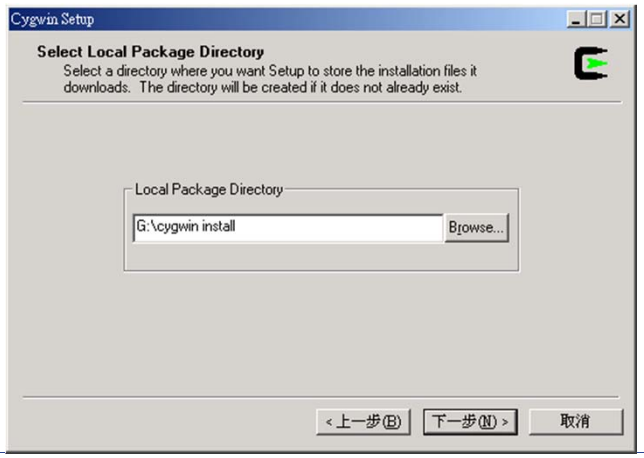
Download Cygwin

- 選擇所要安裝的目的目錄



Download Cygwin

- 選擇安裝時做為儲存下載程式的儲存目錄

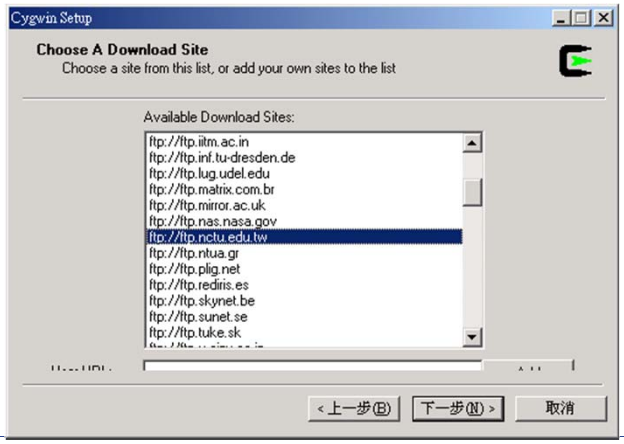


Download Cygwin

- 選擇網路連接的方式

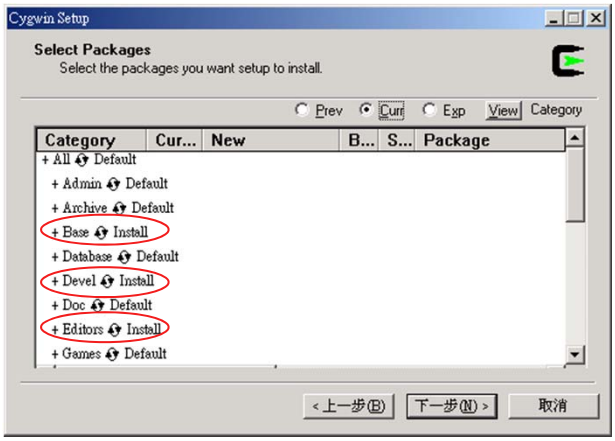
Download Cygwin

- 選擇程式來源網站 (Mirror Sites)



Download Cygwin

- 選擇欲安裝的套件

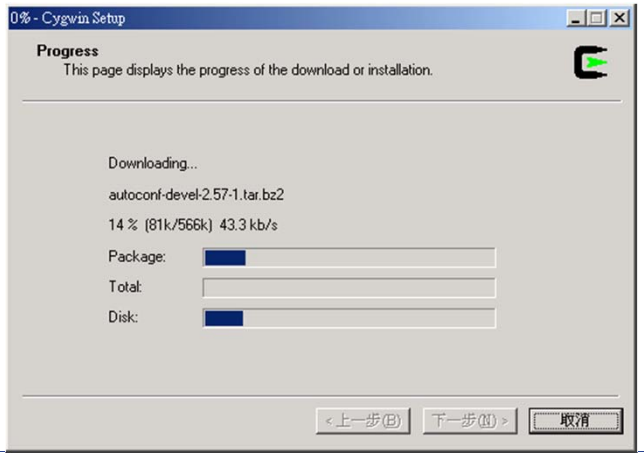


Download Cygwin

- 選擇欲安裝的套件
  - 每點選圖中的雙箭頭圈圈符號時，則會變成“Install”、 “Uninstall”、或 “Default”其中之一的設定
  - 點選加號 “+”則可將該項展開成更細的分項
  - 建議選項包括：
    - Base: 全
    - Devel: 全
    - Editors: vim
    - Math: gnuplot, lapack
    - Utils: rpm, rpm-build
    - Web: wget
    - Lib: 全

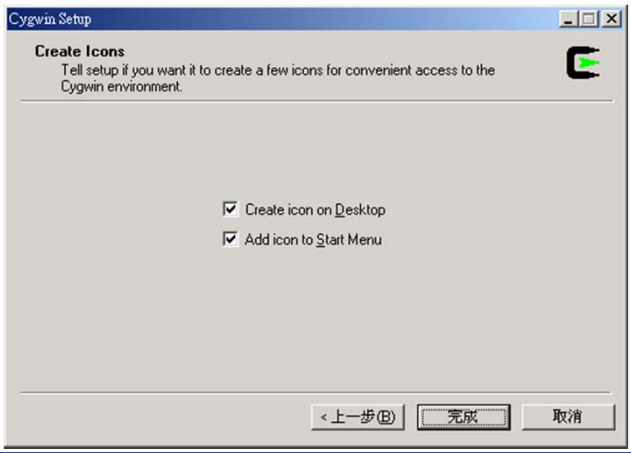
Download Cygwin

- 安裝



Download Cygwin

- 桌面及程式集選項設定



Download Cygwin

- 完成安裝

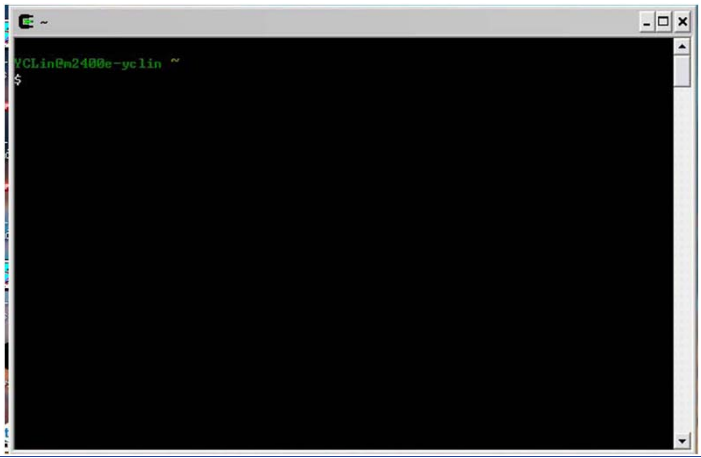
- 如果以後還要安其他套件只要再執行setup.exe,再選擇要安裝的套件即可

移除 Cygwin

- 在 cygwin 當中並沒有自動移除的程式，所以我們要用手動的方式來移除
- Delete the following
  1. cygwin 在桌面以及開始功能表上的捷徑
  2. 啟動 regedit ， 找到 \HKEY_LOCAL_MACHINE\SOFTWARE\Cygnus Solutions 這個 delete 掉就可以了
  3. delete 掉當初 cygwin 安裝的目錄
  4. delete 掉安裝時候所產生的暫存檔

執行Cygwin

- 從程式集中的Cygwin→Cygwin Bash Shell



Cygwin常用指令—Recall

- 文字模式的使用指令
  - ls 列出檔案
  - cp 拷貝檔案 例: cp test1 test2
  - mv 搬移/改名目錄名稱或檔案名稱 例: mv test1 test2
  - cd 更換目錄 例: 要回到一開始的目錄 cd ~/
  - mkdir 創造資料夾 例: mkdir test
  - rmdir 刪除資料夾 例: rmdir test
  - man 說明 例: man mkdir
  - exit 離開

GNU 工具 — C/C++編譯器GCC

- GCC(Gnu Compiler Collection)是由GNU 出的C 語言編譯器， 可將由ANSI C 或traditional C 語言寫成的程式碼編譯成可執行檔
- 由於GCC 能分別編譯出執行於不同硬體、作業系統下的程式， 在Linux 系統上是相當多人用的C 語言編譯程式
- GCC 可以說是Richard Stallman 所創立的GNU 計畫中最重要的作品之一， 它提供了自由軟體世界高品質的編譯器(compiler)
- 如果沒有 GCC，恐怕今日我們也不會有這麼多形形色色的自由軟體可用

GNU 工具 — C/C++編譯器GCC

- 早期 GCC 的發展方向是以 C/C++ 與 Objective C 等語言為主， 故〝GCC〞的函義 即為 〝GNU C Compiler〞
- 但發展到現在， GCC 的內函已不只是 C 與類似 C 的程式 語言而已了， 它同時還包含了許多其他語言的編譯器。這些編譯器 全部共用同一組程式編譯最佳化的引擎， 使用相同的浮點數運算模組，同時也都享有 GCC 高移植性的特色
- 因此，GCC 事實上包含了一個很大的編譯器家族， 而〝GCC〞的函義也隨之轉變為〝GNU Compiler Collection〞

GNU 工具 — C/C++編譯器GCC

- GCC 一個很大的特色是高度可移植性
- 目前已知有超過三十種硬體平台與作業系統 可以執行 GCC， 其中硬體平台包括：
  - x86, ia64, alpha, hppa, m68k, Power PC, mips, IBM rs6000, sparc/sparc64
- 而作業系統則從 Microsoft 平台(DOS/Win32) 到 IBM OS/2 到各家的 UNIX 都有
- 此高度可移植性正是 GCC 廣為流傳散佈的主要原因

GNU 工具 — C/C++編譯器GCC

- 在許多商業版的UNIX 系統中， 如果沒有特別買其專屬的編譯 器的話 (其價格往往不便宜)， 人們通常就選擇安裝 GCC 來使用
- 而且， 儘管GCC 是自由軟體計畫開發出來的， 但其所編譯出來的程式品質並不輸給商業版的編譯器， 甚至在某些平台上所編譯出來程式有更好的執行效能
- 除了各種編譯器以外， GCC 還內含了其他的工具程式以及各程式語言所需的函式庫， 像是 C++ (g++) 所需的libstdc++， 以及Java (gcj)所需的class 函式庫libgcj 等

GNU 工具 — C/C++編譯器GCC

- 其工具程式中， 最重要的就是cpp。它是程式編譯時期的前置處理器
- 通常一個程式在編譯時的步驟如下:
  - 巨集前置處理：preprocess 先處理那些#ifdef #define 這些東西並做一些巨集代換
  - 程式碼編譯與最佳化：compile 做語意分析，翻譯成組合語言
  - 組譯：assemble 翻成機器碼與OS 有關的格式，做成relocatable obj 檔
  - 連結函式庫：link 找到symbol(函式，變數名)與程式庫(shared obj)中的副程式
  - 產生執行檔：做成可執行obj 檔(executable obj)

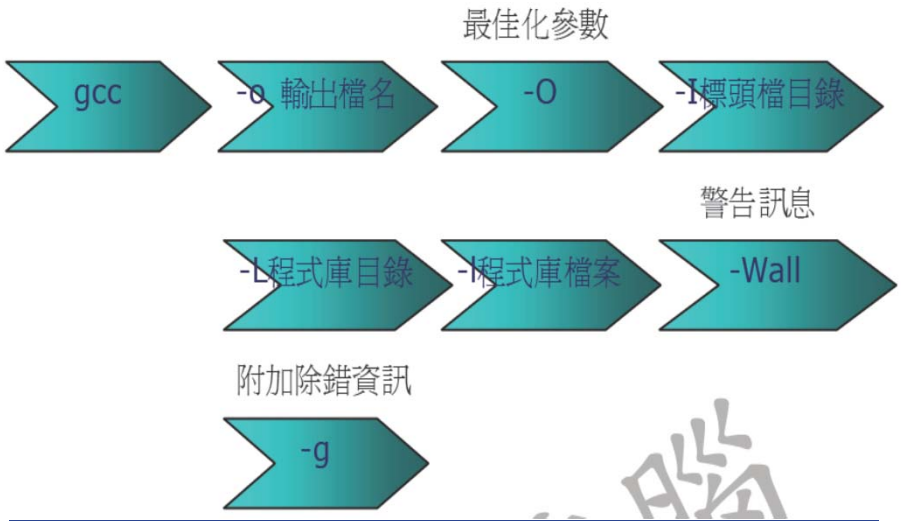
GNU 工具 — C/C++編譯器GCC

- cpp 的工作就是負責整個流程的第一個步驟
- 通常我們在寫程式時會定義許多巨集程式碼，同時也會include 若干「標頭檔」(header files, 附檔名為 .h)
- 這些用以方便程式撰寫的手法在編譯時期都會交由 cpp 處理， 在原始碼中全部展開後， 才交由編譯器進行實際的編譯工作
- 這些在呼叫 GCC 進行編譯時都會自動進行，故我們不需要親自呼叫cpp

GNU 工具 — C/C++編譯器GCC

- 編譯器在編譯過程中， 先將程式碼編譯成 object 檔
- 然後再和程式庫聯結， 成為可執行檔。故一個編譯器須提供的參數主要有幾類：
  - 指定編譯器編出的 object 檔或是可執行檔檔名
  - 在編譯過程做最佳化，可提升程式的執行速度
  - 設定搜尋程式庫的標頭檔 (header file) 及程式庫檔的目錄及指定程式庫檔檔名
  - 提供進一步的資訊以便使用者找尋程式中的錯誤

GCC 的參數設定



GCC 的參數設定

- 上圖表示gcc常用的參數與說明
- 更詳細的參數說明可以與UNIX/Linux命令列輸入
  - gcc –help

```
root@localhost:~
[root@localhost ~]# gcc --help
Usage: gcc [options] file...
Options:
  -pass-exit-codes         Exit with highest error code from a phase
  --help                   Display this information
  --target-help            Display target specific command line options
  (Use '-v --help' to display command line options of sub-processes)
  -dumpspecs               Display all of the built in spec strings
  -dumpversion             Display the version of the compiler
  -dumpmachine             Display the compiler's target processor
  -print-search-dirs       Display the directories in the compiler's search path
  -print-libgcc-file-name  Display the name of the compiler's companion library
  -print-file-name=<lib>   Display the full path to library <lib>
  -print-prog-name=<prog>  Display the full path to compiler component <prog>
  -print-multi-directory   Display the root directory for versions of libgcc
  -print-multi-lib         Display the mapping between command line options and
                           multiple library search directories
```

GCC 的參數設定

- 一些常見的編譯參數， 說明如下：
  - -c: 只建立obj 檔，留待後面才來連結(link)
  - -S: 只建立assembly 檔
  - -g: 建立一些除錯資訊給可執行檔，這樣debug 工具ddd,gdb 才能除錯
  - -o: 把建立的二位元檔給另外名字，因為可執行檔最後內定名字是 a.out
  - -D: 條件編譯，搭配#ifdef #define 用。如果有defined 才編譯
  - -W: 編譯時出錯時，顯示錯誤訊息的條件
  - -L: 給連結時要用到的函式庫的搜尋目錄
  - -I: 標頭檔.h 的搜尋目錄
  - -l: 正常連結只會在libc 這個函數庫，其他函數庫需要用這個指定連結
  - -O1 -O2 -O3: 最佳化，會根據CPU 的架構編出好的程式碼，需要多一點編譯時間

GCC 相關設定說明

- 標頭檔案（header file）
  - 在撰寫程式時， 常常會利用標頭檔案來提供常數的定義、函示宣告等… 。在Linux 系統中一般會將include 的header file 放在/usr/include 目錄下， 在GCC 編譯器下， 可以利用-I 旗標來指定特定的include 路徑， 例如：

    gcc -I/usr/src/include test.c
  - 表示test.c 所用到的include 檔， 可以到/usr/src/include 目錄下找尋

GCC 相關設定說明

- 函示庫（libraries）
  - 函示庫(libraries) 就是一些可再利用且事先編譯的函數， 在Linux 系統中標準的系統函示庫通常放在/lib 或/usr/lib 目錄下
  - 函示庫的名稱大致都以lib 開頭，隨後的名稱就根據該函示庫功能而定，函示庫的型態可以分為下列兩種：
    - a：代表傳統、靜態(static)的函示庫。如：
      - gcc -o test test.c /usr/lib/static_lib.a
    - so：代表共享(shared)函示庫。如：
      - gcc -o test test.c -lm

make 介紹

- 自動化推手
  - 當我們的程式寫得很龐大時 (可能包含許多原始程式檔案)，往往需要相當複雜的編譯動作才能將程式整個編譯完成， 並且安裝到正確的位置上使用
  - 這時候， 光靠手動執行每一個編譯指令顯然是很沒有效率的
  - 同時， 由於一個大型程式的編譯往往要花很久的時間， 故如果每次程式一有修改 (不論修改的幅度有多少)就要整個從頭編譯， 也是相當沒有效率的。
  - 解決問題之道， 就是使用 make 程式，將整個程式編譯的工作自動化

make 介紹

- 自動化推手
  - 當此程式啟動時， 它會讀入目前工作目錄下的指令檔Makefile， 並根據該檔案的指令一步步地執行
  - 因此， 我們就可以將整個程式編譯的細節寫在Makefile 中， 當程式需要編譯時就執行make 將它編譯完成
  - 在 GNU 計畫中， 同樣也提供了一個強大的 make 程式， 它除了具備上述的基本特性之外， 同時還包含了許多 GNU 的延伸功能

make 介紹

- 自動化推手
  - 例如各種巨集的定義， 條件判斷式....等等， 而有些延伸功能是其他版本的 make 所沒有的
  - 因此， 有許多大型的自由軟體， 都會特別指明要GNU make (或者還包括GCC 及其他的GNU編譯工具)才能進行編譯， 因為它們的Makefile 中已隱含了GNU make 許多特有的功能

make 介紹

- makefile 編譯時的規則：
  1.) 如果這個project 沒有編譯過，那麼我們的所有C 檔都要編譯並被連結
  2.) 如果這個project 的某幾個C 檔被修改，那麼我們只編譯被修改的C 檔，並連結目的程式
  3.) 如果這個project 的標頭檔被改變了，那麼我們需要編譯引用了這幾個頭檔的C 檔，並連結目的程式

make 介紹

- Makefile內容結構
  - Makefile檔案內容通常由一組相依規則組成，語法如下：

```
target... : dependencies
<tab>command
<tab>command
target... : dependencies
<tab>command
<tab>command
…
```

  - target 通常是一個檔名，是一個程式的執行檔或目的檔
  - dependencies 則是指檔案間的相依性，說明產生target的相依關係，當dependencies 有異動時，target就不再是最新版本，此時make便會重新編譯，以產生最新的target

make 介紹

- Makefile內容結構
  - 第二部份是指定『規則』，用來描述建立target的方法
  - 規則中的command指的是產生target實須執行的系統指令
  - 建立(輸入)command前，要先按<tab>鍵
- 使用變數
  - 在makefile的規則敘述中，可以自訂一些變數代表一串文字，並使用$(變數名稱)來引用變數所代表的文字串
- all檔名
  - 用來宣告同時編譯多個輸出檔案

TSEG Tamkang University Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

make 介紹

- Makefile內容結構
  - 第二部份是指定『規則』，用來描述建立target的方法
  - 規則中的command指的是產生target實須執行的系統指令
  - 建立(輸入)command前，要先按<tab>鍵
- 使用變數
  - 在makefile的規則敘述中，可以自訂一些變數代表一串文字，並使用$(變數名稱)來引用變數所代表的文字串
- all檔名
  - 用來宣告同時編譯多個輸出檔案

make 介紹

- 實例說明：
  - 假設我們有一個程式， 共分為下面的部份：
    - menu.c 主要的程式碼部份
    - menu.h menu.c 的include file
    - utils.c 提供menu.c 呼叫的一些function calls
    - utils.h utils.c 的include file
  - 同時本程式亦呼叫了ncurses 的function calls。而 menu.c 和utils.c 皆放在/usr/src/menu 下。但menu.h 和utils.h 卻放在/usr/src/menu/include 下。
  - 而程式做完之後， 執行檔名為menu 且要放在 /usr/bin 下面

make 介紹

- Makefile內容如下：

```
# This is the Makefile of menu, filename is makefile01
CC = gcc
CFLAGS = -DDEBUG -c
LIBS = -lncurses
INCLUDE = -I/usr/src/menu/include
all: clean install
install: menu
        chmod 750 menu
        cp menu /usr/bin
menu: menu.o
        $(CC) -o $@ $? $(LIBS)
menu.o:
        $(CC) $(CFLAGS) -o $@ menu.c $(INCLUDE)
utils.o:
        $(CC) $(CFLAGS) -o $@ utils.c $(INCLUDE)
clean:
        -rm *.o
        -rm *~
```

make 介紹

- 在上述的Makefile 中， 要使用某個macro 可用 $(macro_name)如此的形式，make 會自動的加以展開
- $@為該rule 的Target，而$?則為該rule 的depend
- 若在command 的前面加一個&quot-“， 表示若此 command 發生錯誤則不予理會， 繼續執行下去
- make的執行
  - make –f makefile01
  - -f是告訴make指令，使用哪一個makefile

Richard Stallman何許人也？

- **Richard Matthew Stallman** (born March 16, 1953), often abbreviated "**rms**
- He is an American software freedom activist, hacker (programmer), and software developer
- In September 1983, he launched the GNU Project to create a free Unix-like operating system, and has been the project's lead architect and organizer
- With the launch of the GNU Project, he started the free software movement and, in October 1985, set up the Free Software Foundation

Embedded Linux C 程式設計

- 建立靜態函式庫
- 同時Compile多個程式
- 範例程式

mymin.h

```
/* mymin.h */
int min(int x, int y);
```

mymin.c

```
/* mymin.c */
int min(int x, int y)
{
   if (x < y)  return x;
   else return y;
}
```

ex_min.c

```
/* ex_min.c */
#include <stdio.h>
#include "mymin.h"
int main(){
{
   int n1, n2, n3;
   printf("Input first integer number: ");
   scanf("%d", &n1);
            printf("Input first integer number: ");
    scanf("%d", &n2);
    n3 = min(n1, n2);
    printf("The smaller number is %d\n", n3);
     return 0;
}
```


Embedded Linux C 程式設計

- Compiler命令
  - gcc ex_min.c mymin.c –o exmin
- 將二個原始程式檔ex_min.c及mymin.c編譯成一個執行檔exmin.exe

Embedded Linux C 程式設計

- 常用的系統呼叫函式(System Call Functions)
- 程序函式
  - getpid()
  - getppid()
  - exec() Family
  - system()
  - fork()
  - exit()

Embedded Linux C 程式設計

- 參考資訊
- http://www.jollen.org/blog/2006/10/linux_26_system_call12.html
- Access Date: 2010.12.01

Embedded Linux C 程式設計

- getpid() and getppid()

  #include <unistd.h>

  pid_t getpid(void);

  - It will return the process ID of the current process

  pid_t getppid(void);

  - It will return the Parent process ID of the current process

Embedded Linux C 程式設計

- exec() Family
  - There are six functions in the exec() family
  - They are
    - execl(const char *path, const char *arg(), …, (char *)NULL);
    - execlp(const char *file, const char *arg(), …, (char *)NULL) ;
    - execle(const char *path, const char *arg(), const char *envp[]) ;
    - execv(const char *path, const char *argv[]) ;
    - execvp(const char *file, const char *argv[]) ;
    - execve(cons char *path, const char *argv[], const char *envp[]);
      - They will return -1 if fail, No any return if success

Embedded Linux C 程式設計

- system()

  #include <stdlib.h>

  int system(const char *String);

  - It will call execve() function by parameter /usr/sh –c String
  - It will return 127 if no String existed in /usr/sh
  - It will return -1 if the other reasons to fail
  - Return 0 if success

Embedded Linux C 程式設計

- fork()

  #include <sys/types.h>

  #include <unistd.h>

  pid_t fork(void);

  - It will return PID of child process in the original parent process, if success. And it return 0 to the created child process.
  - It will return -1 in the parent process if fail.

Embedded Linux C 程式設計

- exit()

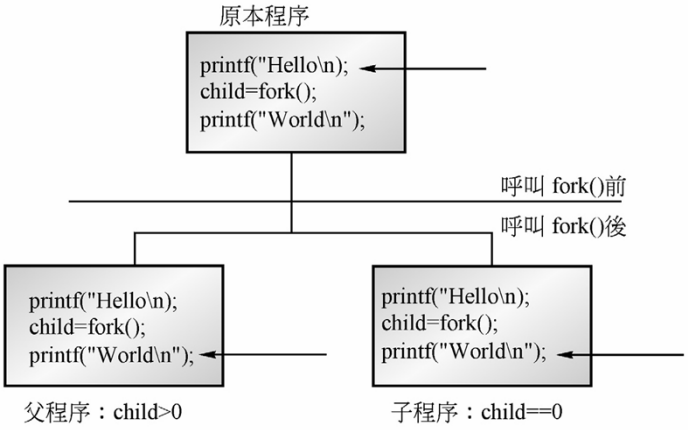
  #include <stdlib.h>

  void exit(int status);

  - It will terminate the execution of this process
  - Status uses to present the terminated status. In general, 0 is normal termination and Non-zero value to present one kind of Informal termination

Embedded Linux C 程式設計

- fork()的概念



Embedded Linux C 程式設計

- zombie程序
  - 過度程序
  - 是指已經結束執行，但是尚未被清除、未釋放記憶體資源的程序
- wait()與waitpid()函式可用來協助Parent process自動清除zombie程序

```
#include <sys/types.h>
#include <sys/wait.h>
pid_t wait(int *status);
pid_waitpid(pid_t pid, int *status, int options)
```


Embedded Linux C 程式設計

- Linux程式常用的Signal 函式
  - signal()
  - alarm()
  - kill()
  - pause()

Embedded Linux C 程式設計

- signal()

  #include <signal.h>

  void (*signal(int sig, void (*func)(int)))(int);

  - sig is one kind of signal, except SIGKILL and SIGSTOP
  - func is the function indicator which will handle the signal, sig.
  - func also be the following constant value
    - SIG_IGN: it means ignore this signal, sig
    - SIF_DFL: it means recover the default handling function for this signal, sig

Embedded Linux C 程式設計

- alarm()

  #include <unistd.h>

  unsigned int alam(unsigned int seconds);

  - It will set an alarm to wait some seconds and generate one SIGALRM signal to specific process.
  - It will return the remaining seconds to generate the signal.

Embedded Linux C 程式設計

- alarm()

  #include <unistd.h>

  unsigned int alam(unsigned int seconds);

  - It will set an alarm to wait some seconds and generate one SIGALRM signal to specific process.
  - It will return the remaining seconds to generate the signal.

Embedded Linux C 程式設計

- kill()
- pause()

GUN gdb Debugger

- GDB: The GNU Project Debugger
  - the GNU Project debugger, allows you to see what is going on `inside' another program while it executes
  - or what another program was doing at the moment it crashed
- GDB can do four main kinds of things to help you catch bugs in the act:
  - Start your program, specifying anything that might affect its behavior.
  - Make your program stop on specified conditions.
  - Examine what has happened, when your program has stopped.
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

GUN gdb Debugger

- The program being debugged can be written in Ada, C, C++, Objective-C, Pascal (and many other languages).
- Those programs might be executing on the same machine as GDB (native) or on another machine (remote).
- GDB can run on most popular UNIX and Microsoft Windows variants.
- Current status, **GDB version 6.8**
- Download GDB from Project GNU's FTP server, or Red Hat's sources site:
  - http://ftp.gnu.org/gnu/gdb (mirrors)
  - ftp://sourceware.org/pub/gdb/releases/ (mirrors).

GUN gdb Debugger

- 命令列使用的範例

  gdb prog.out 說明：debug prog.out 這個執行檔

  gdb > run 說明：在gdb下開始執行

- 了解gdb的使用
  - gdb –help
- 離開gdb
  - quit [*expression*]
  - q
    - To exit GDB, use the quit command (abbreviated q), or type an end-of-file character (usually Ctrl-d).

GUN gdb Debugger

- gdb –help

(gdb) help List of classes of commands:
aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands
Type "help" followed by a class name for a list of commands in that class.
Type "help" followed by command name for full documentation.
Command name abbreviations are allowed if unambiguous. (gdb)

Outline

- Introduction to Embedded Systems
- Understanding the ARM serious
- Famous Embedded OS
- Linux Overview
- Introduction to Linux Programming
- **Guide to Programming on ARM serious**
- Embedded Systems Development

Guide to Programming on ARM serious

- Download the Guidebook
  - TKUEmbeddedDevelopmentSystem
- 熟悉arm 嵌入式系統平台與發展環境
- IP port: **163.13.128.2xx**
- 帳號：自己的學號
- 密碼：6666，第一次登錄後，可用passwd只令修改

Guide to Programming on ARM serious

- 如何於PC Linux 建立嵌入式系統之發展環境?
  1. 下載cross compiler(toolchain)，
  2. 安裝cross compiler，
  3. 使用cross compiler 產生執行檔，
  4. 利用sftp 上傳執行檔到目標板，
  5. 利用ssh 連線到目標板進行程式測試。
- 整個過程是透過網路進行，學生的學習場地可以不限於教室，也可以隨時選擇適合的時間學習

Guide to Programming on ARM serious

- **實驗器材**
  - PC：PentiumIII以上、128M、寬頻上網
  - 華亨QT2440(或以上)實驗版：400MHz、64M Flash、64M DRAM
  - 軟體：
    - PC安裝Linux、ARM CROSS Compiler
    - QT2440安裝ssh、sftp

Guide to Programming on ARM serious

- 基本作業程序
  - 下載toolchain(即cross compiler)
  - 使用cross compiler產生實驗板的目的碼(執行檔)
  - 使用sftp上傳執行檔到實驗板
  - 使用ssh連線到實驗板進行測試

Outline

- Introduction to Embedded Systems
- Understanding the ARM serious
- Famous Embedded OS
- Linux Overview
- Introduction to Linux Programming
- Guide to Programming on ARM serious
- **Embedded Systems Development**

Embedded Systems Development

- 實驗題目
  - 詳指定作業題目