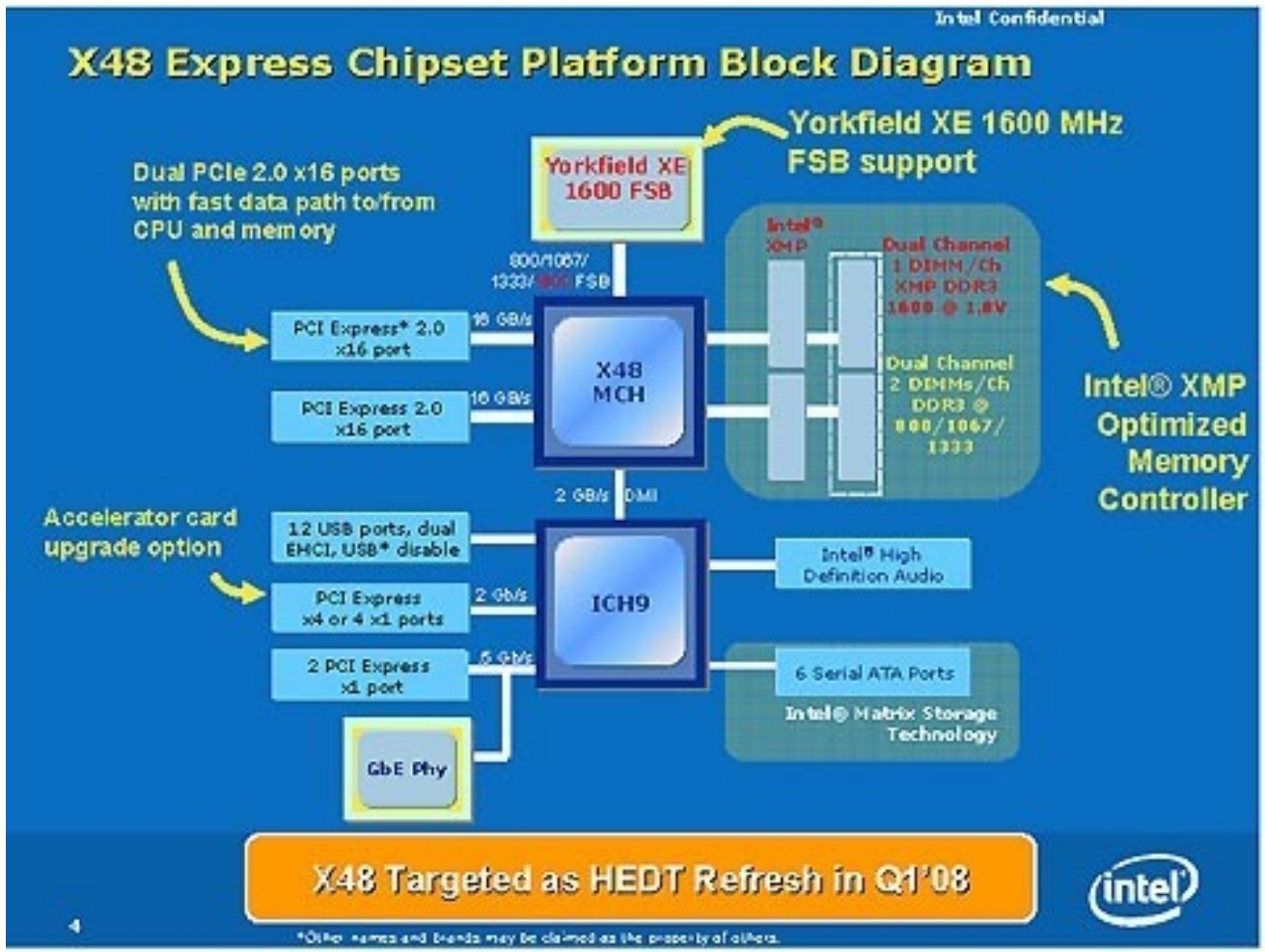
中斷與中斷處理

大綱

- 中斷與中斷處理程式
- Linux核心中斷處理程式的運作
- 中斷處理環境與中斷處理的實作
- 中斷子系統的控制
- Bottom Half與Bottom Halves
- Bottom Halves的選擇與同步鎖定
- 本章重點回顧

主機的硬體配置

- 各元件都有一個位址來定位，稱爲
  - I/O位址
- 各元件與CPU之間的溝通則是Interrupt，稱爲
  - 中斷或岔斷



CPU與周邊設備的溝通

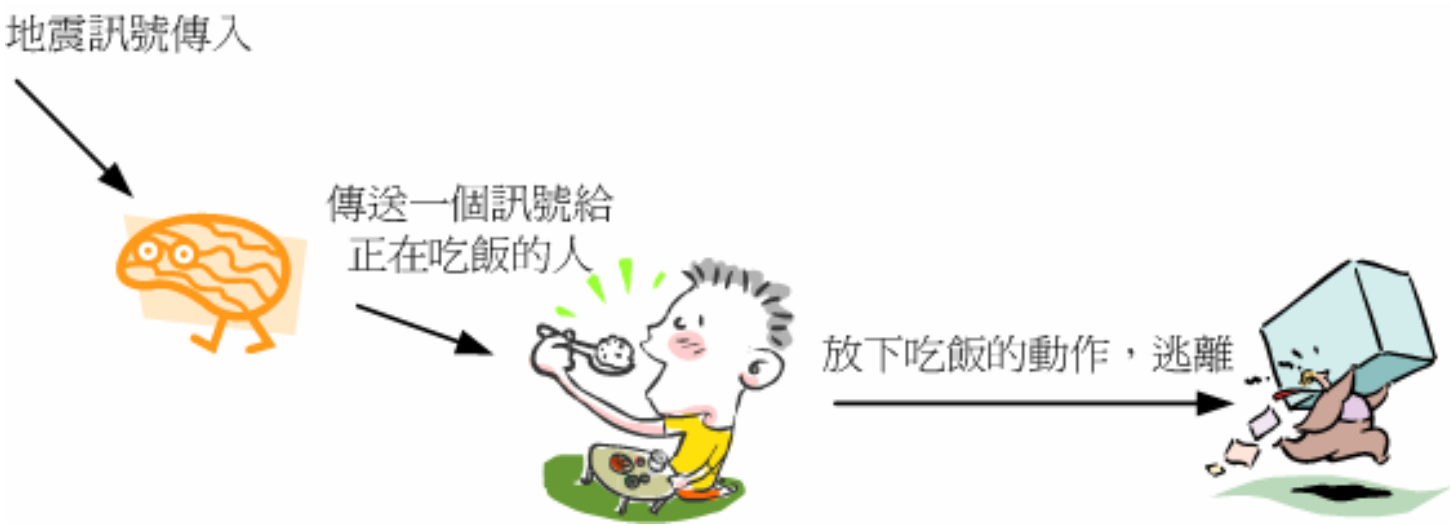
- Linux核心的功能：
  - 管理主機上的各項硬體
  - 與每個裝置進行溝通
- 溝通的方式：
  - 透過定期的方式輪詢(Polling)各項設備
    - 但效能不佳
  - 硬體有資料需求時，提供訊號給核心，讓核心知道目前的周邊硬體有額外的需求產生
    - 就是中斷的方法！

中斷的意義

- 中斷的意義：
  - 允許硬體與處理器進行溝通的一個作業
  - 其實就是訊號的傳遞！

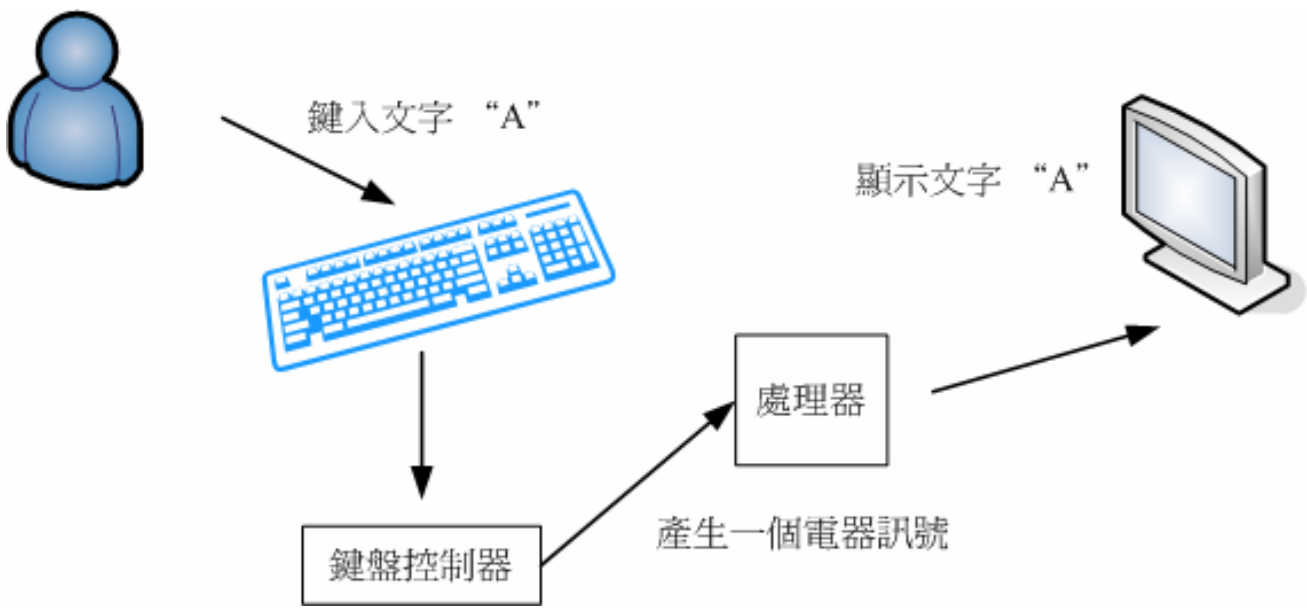
中斷處理的範例—生活範例

- 地震時的情況
  - 訊號：地震
  - CPU：大腦
  - 原先動作：吃飯
  - 中斷後行為：逃難



中斷處理的範例—作業系統

- 輸入文字時的情況
  - 訊號：鍵盤的電器訊號
  - 原先工作：暫停一下下
  - 後來工作：螢幕顯示正確的文字資訊



中斷的特性

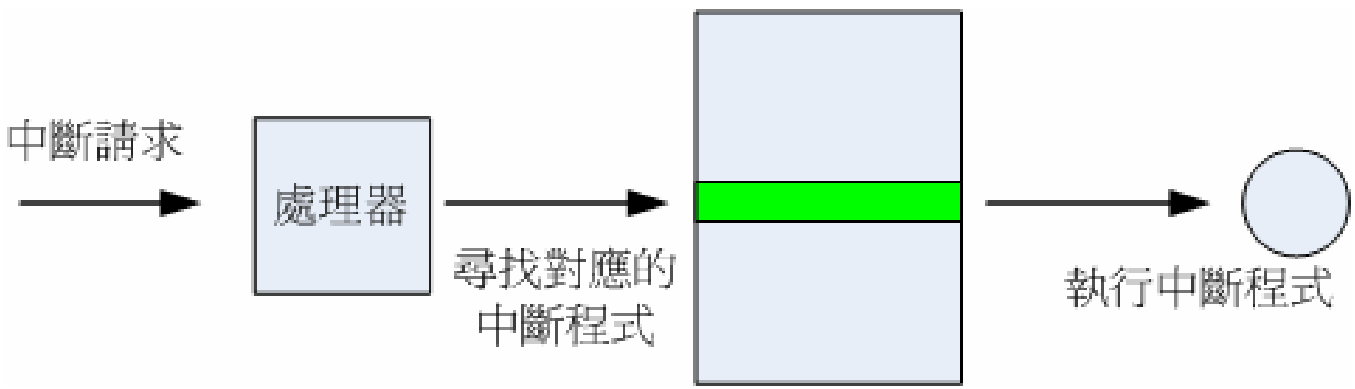
- 非同步產生中斷(任何時刻都可能發生中斷)
  - 因為周邊硬體比CPU慢太多了
  - 核心得以隨時處理中斷
- 系統周邊硬體都有特定的中斷位址
  - 所以作業系統可知道那個周邊目前有中斷的要求
  - 中斷要求線路(IRQ Lines)
    - 固定配置：ex> 電腦計時器、鍵盤控制器
    - 動態配置：ex> PCI匯流排上面的裝置設備

異常狀況(同步式中斷)

- 同步式中斷(Synchronous Interrupt)
  - 異常狀況與處理器時脈同步產生
  - 為CPU發出去通知系統的異常行為
    - 指令執行錯誤
    - 分頁錯誤等

核心處理中斷的方法

- 中斷處理程式(Interrupt handler)
  - 每個中斷設備會與一個中斷處理程式連結
    - 中斷處理程式是驅動程式的一部份
  - 可以隨時讓打斷的程式回復工作



中斷服務的處理

- 中斷服務依屬性切割成兩部份：
  - 前半部處理(Top half)
    - 中斷程式立刻回應硬體或是重置硬體等必要工作
    - 讓硬體可以恢復正常，繼續工作
  - 後半部處理(Bottom half)
    - 等到系統較閒時，再處理後續的任務。
    - 參考下一頁的網路傳輸範例

中斷服務的處理—範例

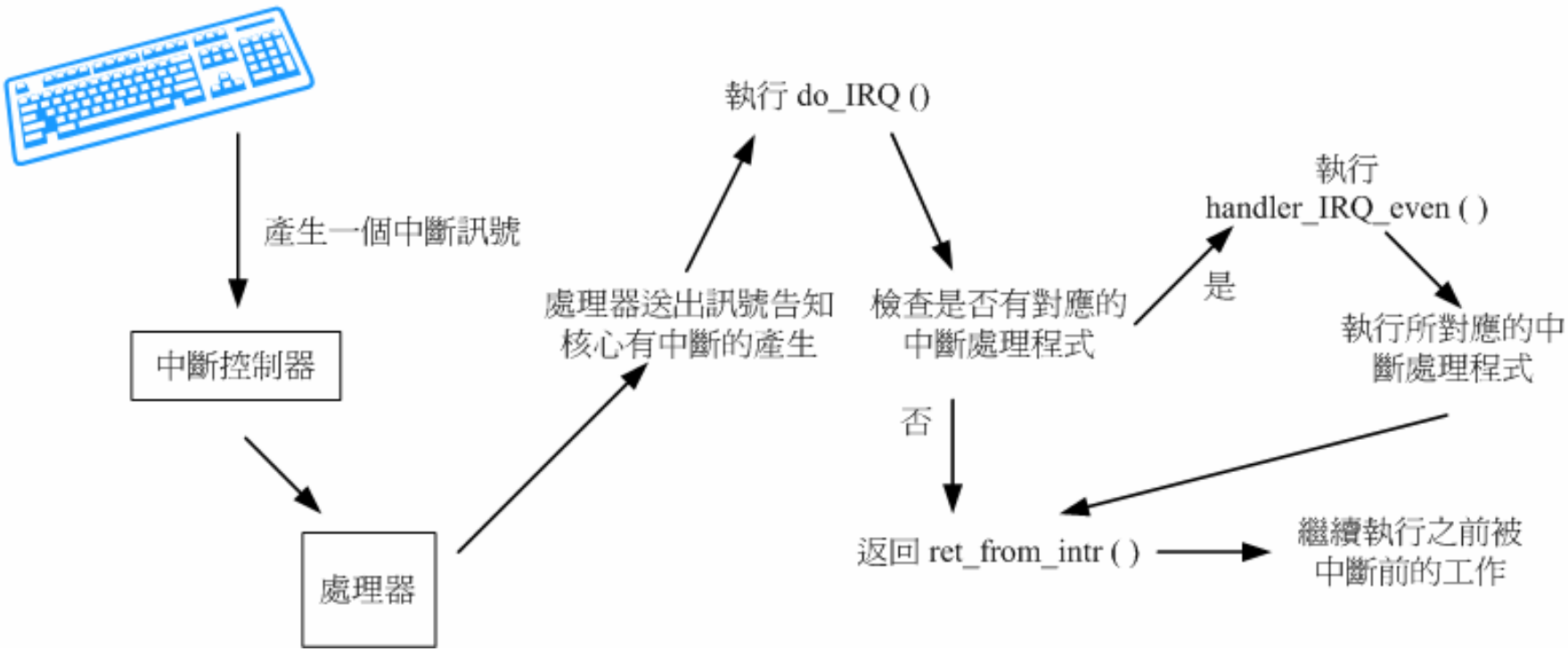
- 網路裝置(如網路卡)傳輸的資料：
  - 前半部處理(Top half)
    - 網路卡收到流入的網路封包，傳送中斷給核心
    - 核心立刻執行網路裝置的中斷程式
    - 將網路卡收到的資料複製到主記憶體中
    - 網路卡恢復正常動作，繼續監聽網路資料
  - 後半部處理(Bottom half)
    - 處理主記憶體內的網路資料

中斷處理時的環境

- 中斷環境的特色
  - 核心並不會與中斷程序有任何的結合動作
    - 此環境中，並沒有程序的概念
  - 中斷環境不會進入休眠模式中
    - 因為無法使用CPU排程器
  - 強調時間概念
    - 減輕對於系統其他部分的負擔

中斷處理的實作

- 鍵盤輸入的中斷處理流程
  - 妳打字時螢幕會立刻有字(因為中斷產生)
  - 原先系統的動作會在後續繼續進行(中斷結束後)



/proc/interrupts

```
[root@hcserver root]# cat /proc/interrupts
           CPU0       CPU1
  0:   97327766    97333993   IO-APIC-edge  timer
  1:          2          1   IO-APIC-edge  keyboard
  2:          0          0   XT-PIC   cascade
  5:          5    1355141   IO-APIC-level  eth1
  8:          0          1   IO-APIC-edge  rtc
 10:         14         16   IO-APIC-level  sym53c8xx
 12:    7417750         31   IO-APIC-level  eth0
 14:    6878662    6943552   IO-APIC-edge  ide0
NMI:          0          0
LOC:  194662831  194662831
ERR:          0
MIS:          0
```

IRQ　發生次數　發生次數　控制器(硬體周邊)

中斷的競爭效應處理

- 核心對中斷的動作
  - 在臨界區間存取資源時，可能會產生競爭效應
  - 為了克服競爭效應，有鎖定的機制
  - 當CPU收到鎖定程序後，可能會關閉某些中斷程式，以避免干擾鎖定的程序
    - 所以系統有很多中斷的子系統控制

中斷子系統的控制

- 中斷控制函式

函式名稱	說明
local_irq_disable( )	取消目前處理器的中斷接收功能
local_irq_enable( )	啟動目前處理器的中斷接收功能
disable_irq( )	關閉指定的中斷線路，於確認線路上相關中斷處理程式完成後才返回
disable_irq_nosync( )	關閉指定的中斷線路
enable_irq( )	重新啟動指定的中斷線路
in_interrupt( )	測試核心目前的執行環境
in_irq( )	測試核心是否正在執行中斷處理程式

Bottom Half與Bottom Halves

- Bottom Half與核心版本現況

核心版本	Bottom Half
2.5版移除	BH
2.5版移除	Task Queues
2.3版移除	Softirqs
2.3版移除	Tasklets
2.5版增加	Work Queues

Bottom Halves的選擇與同步鎖定

- 2.6版核心中，提供三套Bottom Half機制：
  - softirqs
  - tasklets
  - 工作佇列（work queues）

Bottom Half	執行環境	序列執行
Softirqs	中斷處理環境	無
Tasklets	中斷處理環境	有
工作佇列	程序環境	無

本章重點回顧

- 了解何謂中斷與中斷處理程式。
- 了解於Linux作業系統核心內所提供的中斷處理程式機制。
- 了解於Linux作業系統中當發生中斷時核心的運作模式。
- 了解中斷處理子系統的運作方式。
- 了解Bottom Half與Bottom Halves的差異