**直接列出考古題2(精華完整35題含解答)**

**1. C/C++ 類：**

**1)  const int\* p 和 int\* const q 兩者之差別？**

前者: 指到的資料型態為const  
後者: 指標本身為const

**2)   32-bit machine用C語言對數值 0x00005000 的第三個bit設成1，第五個bit設成0**



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #define BIT3 (0x0004)  #define BIT5 (0x0010)    unsigned int a=0x00005000;  void set\_bit3(void) { a |= BIT3;}  void clear\_bit5(void) { a &= ~BIT5;} |

\*延伸閱讀: 若改為針對位址0x00005000, 應改如何改寫? (感謝Joseph提供)



|  |  |
| --- | --- |
| 1  2  3  4 | volatile unsigned int\* a= NULL ;  a = (volatile unsigned int\* ) 0x00005000 ;  void set\_bit3(void) { \*a |= BIT3; }  void clear\_bit5(void) { \*a &= ~BIT5; } |

**3) 指標與陣列的差別？**

宣告陣列的時候，會同時宣告記憶體位址供陣列使用；宣告指標時，則不會。

**4) 試寫出一個Macro求出兩數之最大值。**



|  |  |
| --- | --- |
| 1 | #define MAX(A,B) ( (A) >= (B) )? (A) :(B) ) |

**5) #define SUM(a,b) a+b , 若是 SUM(2,5)\*10 的答案是什麼？**



|  |  |
| --- | --- |
| 1 | SUM(2,5)\*10 = 2+5\*10 = 52 |

**6) 給予10個任意整數，輸出其最小值、最大值、平均值。**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int Max = a[0], Min = a[0], Avg = 0;      for(int i = 0 ; i < 10 ; i++)      {          if(Min > a[i])              Min = a[i];          if(Max < a[i])              Max = a[i];          Avg += a[i];      }     cout << Min << Max << static\_cast(Avg)/10 << endl; |

**7) 說明並解釋下列之interrupt service routine 之錯誤處？**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | //interrupt service routine  \_\_interrupt double isr(double r)  {  double area = PI\*r\*r ;  printf("%f\n",area) ;  return area ;  } |

Ans:  
ISR不能有返回值(不知道給誰)  
ISR不能傳遞參數(不知道誰呼叫)  
浮點運算和pintf有重入問題, ISR應短而高效

**8) 寫出一個字串拷貝程式： void StrCpy(char\* dst , char\* src) ;**



|  |  |
| --- | --- |
| 1 | for(; \*src != '\0', \*dst = \*src; dst++, src++) |

**9) 寫出整數轉換字串程式**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #include "stdio.h"  void int2str(int , char \*);    int main() {    int i = 123;    char s[64];    int2str(i, s);    puts(s);  }    void int2str(int i, char \*s) {    sprintf(s,"%d",i);  } |

**10) 寫出一個程式若輸入為 12345678 , 則返回值為 56781234 DWORD fun(DWORD num)**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | typedef unsigned long DWORD;  DWORD fun(DWORD num)  {      DWORD h, l;      h = num << 16;      l = num >> 16;      return h ^ l;  } |

備註: DWORD 就是 Double Word，每個word为2個Byte的長度，DWORD 即為4個Byte，每個Byte是8位bits，共32位bits

**11) 若x=456;則return值為多少**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | int fun(int x)  {     int count = 0 ;     while(x){        count++ ;        x = x & (x-1) ;     }     return count ;  } |

Ans: 4

**12) 連續呼叫 func 10 次，印出的值為何？**



|  |  |
| --- | --- |
| 1  2  3  4  5 | void func(void){     static int i = 0 ;     i++ ;     printf("%d " , i );  } |

ans: 1 2 3 4 5 6 7 8 9 10

**13) 何謂this指標？何謂template? 何謂virtual function?**

– this是一個被C++保留的詞彙，在某個class裡面指向自己這個物件的指標，儲存自己的記憶體位置

– C++ 的 Template 是種將資料型態參數化的功能。將資料型態資訊自程式碼中抽離，代之以簡化的符號 (T, T1, T2, …)。再由編譯器透過類似巨集代換的方式，根據樣板內容產生實際的程式碼。([參考](http://blog.roodo.com/rocksaying/archives/3641717.html))

– 虛擬函數就是指一個函數的行為可以在其所屬類別之衍生類別中被另一個和該函數具有相同簽章(signature) 之同名函數所重新設計和替換掉. 換句話說, 虛擬函數存在的目的就是讓衍生類別可以自行設計修改原有之函數行為. 如. Function Template (函數樣板) 與 Class Template (類別樣板)。 ([參考](http://ublearning.blogspot.tw/p/virtual-function.html))

**14) 寫出一個程式輸入幾點幾分，return 值為時針與分針的角度(需注意若為9:00則其角度為90度，非270度)**

這個題目的輸入格式是 HH:MM 的格式，我們還是可以直接用 scanf(“%d:%d”, &h, &m) 的方式還讀取輸入的資料。接下來計算完時針與分針各別的角度求兩者的差，並用下面的規則處理：

如果這個值小於 0，則將它的負號拿掉。  
如果這個值大於 360，則將它減去 360。  
如果這個值大於 180，則以 360 減去該值。(由於取的是小的角度)  
我們把這些想法寫成程式：



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include<stdio.h>  void main()  {     int h, m;     float dh, dm, d;       scanf("%d:%d", &h, &m);     dm = m\*6;     dh = h\*30 + m\*0.5;     d = dh-dm;     if(d <0) d = -d;     if(d >=360) d=d-360;     if(d >=180) d=360-d;     printf("%.3f\n", d);  } |

**15) 撰寫一程式，由鍵盤輸入一個整數，然後判別此數是否為質數(prime)**

(質數是指除了1和它本身之外，沒有其他的數可以整除它的數，例如，2,3,5,7與11等皆為質數)



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | #include<stdio.h>  int isPrime(int n);  int main()  {      int n;      scanf("%d", &n);      if(isPrime(n))      {          printf("It's a prime\n");      }else{          printf("It's not a prime\n");      }      return 0;  }  int isPrime(int n)  {      if(n==1)          return 0;      int i=2;      for(; i\*i<=n; i++)      {          if(n%i==0)          {              return 0;          }      }      return 1;  } |

我們不需要找所有的因數，只需要找一半的因數就行了  
例如: 16的因數:1, 2, 4, 8, 16 所以程式跑到i=4的時候，i\*i=4\*4<=16，迴圈就結束了

**2. OS類：**

**1) 何謂reentrant程式，設計reentrant需注意什麼？**

Reentrant function 就是可以被重疊執行的 function，講白話就是一個動作還沒做完之前，又重新做了同樣的動作，重疊執行的狀況發生卻不會影響到結果的正確性，符合這種特性即稱為 Reentrant

有些情況就要把 function 設計為 Reentrant，例如多個 Thread 執行同一個 function，或是在 Thread 中在做了某些事後又呼叫自己，整數比大小就是個 Reentrant function 的例子



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int Max(int a, int b)  {     if(a > b)     {       return a;     }       return b;  } |

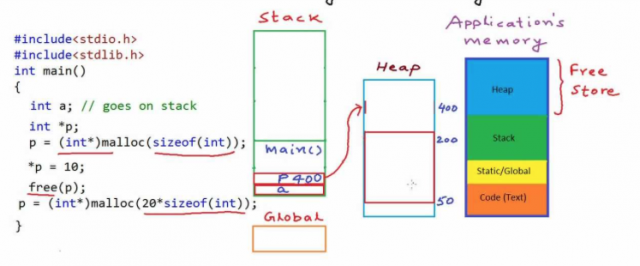
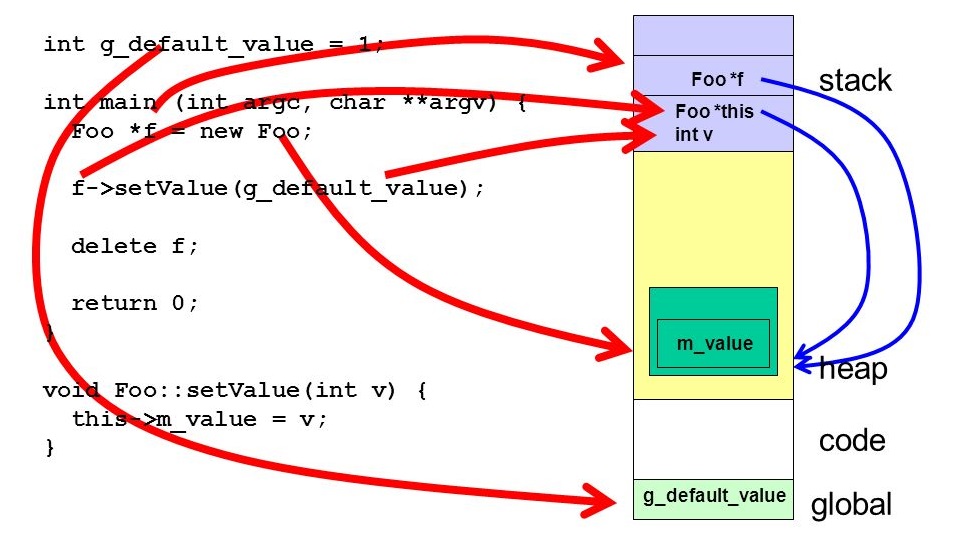
注意這種 function 的條件為  
1. 非 const 的區域或全域變數不可使用 (以免之後的 Thread 改變了值)  
2. return 值不可為非 const 的區域或全域變數(理由同上)  
3. 僅依賴呼叫者提供的參數(安全又衛生)  
4. 可重疊執行，所以沒有 Lock 的機制(很直觀)  
5. function 內部呼叫的 function 也必須為 Reentrant (不然呢？)

**2) 解釋global, stack與heap:**

**global:**  
用來放全域變數、靜態變數(static)等等。

**stack:**  
台灣正體中文稱為堆疊，大陸叫做棧。 Stack中常見的存放資訊如下：區域變數(local variable)、函式參數(function/method parameter)、函數的返回位址(function/method return address)等資訊。 可預測性外加後進先出的生存模式，令stack無疑是最佳的存放策略。 所以必須在**編譯時期為已知**。 這些變數的回收會發生在它從堆疊pop出去的時候，因為個數、大小什麼的都是已知，所以系統知道怎麼進行配置與回收。 stack中的資料之存活週期規律故由**系統自行產生與回收其空間**即可，就不勞工程師們費心啦！

**heap:**  
台灣正體中文稱為堆積，大陸叫做堆。 不可預測的因素造成上述的stack區塊不適合運用。所以當資訊為動態配置產生，系統會存放在另外一塊空間，稱之為『Heap』(注意這裡的Heap跟資料結構中的Heap不相關，可別會錯意！)。 這裡的記憶體由使用者負責進行回收，配置則是由malloc或是new來負責。 所以使用Heap記憶體主要是**編譯時期還不知道** 大小或個數的變數。 例如說，你需要用一個陣列，這個陣列的大小要在執行的時候由使用者的輸入來決定，那你就只能使用動態配置，也就是把這個陣列配置在heap中。 Heap中的資料如果沒有正常的回收，將會逐步成長到將記憶體消耗殆盡，下次發生上述問題的實後，**切記自己檢查一下heap空間的資料有無正常回收**。

**程式碼與記憶體中的配置狀況可以參考這些圖:**  
**實際範例1:**  
全域變數以及Code/TEXT程式碼會存在GLOBAL區, 而編譯過程已知的部分就放到STACK區, 那麼RUN TIME時期動態配置的空間就在HEAP區囉!  
[](https://2.bp.blogspot.com/-2MNFF69FSR8/Wl3O7CgY5xI/AAAAAAAAI58/swf2nCeisXgiNJWNGCbGEyffYwyr_fcGgCLcBGAs/s1600/stack-heap.png)  
**實際範例2:**  
如圖記憶體的分布, 變數與function pointer都被先分配到STACK, 若有動態配置空間會在HEAP區, 而Global區就是全域變數以及Code/TEXT程式碼也存在這裡!  
[](https://1.bp.blogspot.com/-yUY-UEGfBEE/Wl3QQm0axgI/AAAAAAAAI6M/XqFai4-Zg6oNdkAhq6bEvz4lxhEjOLsTACLcBGAs/s1600/C%252B%252BMemory.jpg)

**3) 何謂deadlock?**

一組processes陷入互相等待的情況(Circular waiting), 造成processes接無法往下執行，使得CPU utilization及Throughput大幅降低

**Deadlock成立的四個必要條件:**  
1. Mutual exclusion(互斥)  
2. Hold & wait(持有並等待) (Partial Allocation)  
3. No preemption(不可強取豪奪)  
4. Circular waiting(循環等待)

**4) 說明 mutex 與 semaphore**

Mutex是一把鑰匙，一個人拿了就可進入一個房間，出來的時候把鑰匙交給隊列的第一個。一般的用法是用於串行化對critical section代碼的訪問，保證這段代碼不會被並行的運行。  
(A mutex is really a semaphore with value 1.)

Semaphore是一件可以容納N人的房間，如果人不滿就可以進去，如果人滿了，就要等待有人出來。對於N=1的情況，稱為binary semaphore。一般的用法是，用於限制對於某一資源的同時訪問。

**Binary semaphore與Mutex的差異：**  
在有的系統中Binary semaphore與Mutex是沒有差異的。在有的系統上，主要的差異是mutex一定要由獲得鎖的process來釋放。而semaphore可以由其它process釋放（這時的semaphore實際就是個原子的變量，大家可以加或減），因此semaphore可以用於process間同步。 Semaphore的同步功能是所有系統都支持的，而Mutex能否由其他process釋放則未定，因此建議mutex只用於保護critical section。而semaphore則用於保護某變量，或者同步。

**另一個概念是spin lock，這是一個內核態概念:**  
spin lock與semaphore的主要區別是spin lock是busy waiting，而semaphore是sleep。對於可以sleep的process來說，busy waiting當然沒有意義。對於單CPU的系統，busy waiting當然更沒意義（沒有CPU可以釋放鎖）。因此，只有多CPU的內核態非process空間，才會用到spin lock。 Linux kernel的spin lock在非SMP的情況下，只是關irq，沒有別的操作，用於確保該段程序的運行不會被打斷。其實也就是類似mutex的作用，串行化對critical section的訪問。但是mutex不能保護中斷的打斷，也不能在中斷處理程序中被調用。而spin lock也一般沒有必要用於可以sleep的process空間。

**5 設計OS的重點在哪些？**

作業系統的主要任務是將底層的硬體抽象化，使得應用軟體能以更方便、更有效率、更安全的方式使用硬體資源。換言之，硬體效能是否能徹底被發揮，作業系統的設計扮演非常重要且關鍵的角色。作業系統是無所不在的，它存在於大至企業等級的伺服器，小至個人的桌上型電腦，以及最近火熱的智慧型手機。在不同的環境之下，硬體資源也不同，因此作業系統的設計也有所不同。

舉例來說，在大型的伺服器中，因為硬體資源豐沛，故作業系統的設計重點可能在穩定度以及安全性方面。在嵌入式系統中，因為記憶體以及電源都很拮据，故作業系統的主要議題可能在能耗與記憶體的節省上。而在即時系統中，由於計算工作必須在預設的時間之前完成，因此作業系統必需聰明地安排工作，確保它們能及時完成。

**6) 如何 Linux 與 windows 互相傳送檔案？**

samba，是種用來讓UNIX系列的作業系統與微軟Windows作業系統的SMB/CIFS（Server Message Block/Common Internet File System）網路協定做連結的自由軟體。

**7) 何謂DLL?**

Dynamic Link Library(動態連結檔)，就是將一些function封裝起來,以達到資源共用、程式共用的目的, 一方面也能節省記憶體空間…

**8) uClinux 與 Linux 最大差異在哪？**

uClinux 就是Micro-Control-Linux，字面上的理解就是”針對微控制領域而設計的Linux 系統，沒有MMU支持是uClinux與主流Linux的基本差異。標準Linux是針對有MMU的處理器設計的。在這種處理器上，虛擬地址被送到MMU，把虛擬地址映射為物理地址。通過賦予每個任務不同的虛擬-物理地址轉換映射，支持不同任務之間的保護。

對uCLinux 來說，其設計針對沒有MMU的處理器，不能使用處理器的虛擬內存管理技術。uCLinux仍然采用存儲器的分頁管理，係統在啟動時把實際存儲器進行分頁。在加載應用程序時程序分頁加載。但是由於沒有MMU管理，所以實際上uCLinux采用實存儲器管理策略。uCLinux係統對於內存的訪問是直接的，所有程序中訪問的地址都是實際的物理地址。操作係統對內存空間沒有保護，各個process實際上共享一個運行空間。一個process在執行前，係統必須為process分配足夠的連續地址空間，然後全部載入主存儲器的連續空間中。

**9) 何謂即時多工系統？**

**即時:**  
即時的意義是指當某事件發生時，必須在一定的時間內，將事件處理完成。

**即時多工:**  
限制完成時間的多工方式是即時多工，支援即時的作業系統叫做即時多工作業系統。即時多工作業系統又分軟即時多工作業系統與硬即時多工作業系統，他們的差異在於對即時的要求程度不一樣，並不是每個即時多工作業系統都可以達到最高的即時水準。

**每個作業系統都有它瞄準的戰場，不能期望它的每個方面都完美:**  
大的作業系統所面對的環境比較複雜，往往做不到即時性，把每份工作穩當的完成是它的主要工作，如 WINDOWS、LINUX。小的作業系統所面 對的環境比較單純，即時性較佳，所以市面上大多數的即時作業系統通常都比較小，如 ECOS、UC/OS-II、VXWORKS。

**3. 計組、硬體類：**

**1) 何謂DMA，有何好處？**

DMA（Direct Memory Address）  
使用DMA存取的話, 硬碟運作時比較不會透過 CPU 運算，因為電腦與周邊設備的傳輸一般透過兩種方式：  
一是透過 CPU 控制來進行資料的傳送；二是在專門的晶片控制下進行資料的傳送。

而我們所說的DMA，就是不用透過 CPU 控制，周邊設備會跟記憶體之間相互傳送資料的通道，在這種方式下，周邊設備利用 DMA 通道直接將資料寫入 HD 或將資料從 HD 中讀出，而不用經過 CPU 運算，系統的速度會因此增加。

**2) 何謂Little endian / Big endian**

endian指的是當物理上的最小單元比邏輯上的最小單元小時，邏輯單元對映到物理單元的排布關係。

如果你在文件上看到一個雙字組的data，  
Ex: long MyData = 0x12345678，要寫到從0x0000開始的記憶體位址時。

如果是Big Endian的系統，  
存到記憶體會變成 0x12 0x34 0x56 0x78，最高位元組在位址最低位元，最低位元組在位址最高位元，依次排列。  
如果是Little Endian的系統，  
存到記憶體會變成 0x78 0x56 0x34 0x12，最低位元組在最低位元，最高位元組在最高位元，反序排列。

比較的結果就是這樣：  
big-endian little-endian  
0x0000 0x12 0x78  
0x0001 0x34 0x56  
0x0002 0x56 0x34  
0x0003 0x78 0x12

**3) 何謂 JTAG? 何謂ICE？**

JTAG是聯合測試工作群組（Joint Test Action Group）,何謂JTAG測試，簡言之，係順序掃描積體電路元件之全部外界輸入輸出腳端，擷取輸入輸出端之測試數據，藉以測試元件內部之功能，或進行被安裝在印刷電路基板上之測試方法。  
ICE是在線模擬器 (In-Circuit Emulator, ICE) 是用來除錯嵌入式系統軟體的硬體設備

**4) 解釋 write back 與 write through**

write-through cache:  
使用 write-through 的 cache,,資料寫入cache時也會同步寫入儲存裝置

write-back:  
而 write-back 是將資料量儲存到一定的量之後,會依據同區塊的資料一次整批寫回去.裡面有提到 dirty ,他是在記憶體裡面 cache 的一個 bit 用來指示這筆資料已經被 CPU 修改過但是尚未回寫到儲存裝置中.

**5) 列舉幾個serial port, parallel port**

serial port :  
終端（Computer terminal）/ 數據機/舊式串列埠滑鼠

parallel port:  
印表機 / 影像掃描器 / 並列裝置，如EPROM編程器

**6) 說明 Watchdog 之運作機制**

Watch Dog, 就是一隻程式或一個系統, 去到一個或各個系統去巡邏, 看是否出問題, 而當出問題時, 將會啟動一些備援機制, 或者是被一些系統的 Disable, 或相對應的行為, 這就是 Watch Dog 系統

**4. 網路類：**

**1) 分別說明switch、hub、router、gateway**

**1.HUB集線器：**  
將所有線路全部連結起來，除了廣播電子訊號之外，並不做任何處理。  
OSI模型層級:L1實體層

我們稱做集線器，一般而言，HUB有兩大特性，一個就是廣播，一個就是半雙工。  
廣播是指，當A電腦要透過HUB送資料給B電腦的時候，A送出來的資料其實連接在這台HUB上的電腦都會收到，但是只有B電腦會將資料收起來，其他電腦則是將封包丟掉。  
半雙工是指，收資料或送資料不能同時，你一次只能做其中一種。  
由於HUB的這種特性，所以當HUB連接非常多電腦時，網路就會變慢。

**2.Switch交換器:**   
藉由MAC位址判斷訊框應該往哪裡送。  
OSI模型層級 L2資料連結層

中文叫交換器，和HUB看起來一樣，但實際上差別很大。首先switch並不一直廣播，而且是全雙工的。主要是SWITCH會記錄封包中的MAC位址所以當電腦A傳送資料給電腦B時，其他電腦並不會也收到資料，而且這個時候別的電腦也可以同時互相傳送資料。雖然SWITCH有上述的好處，但是要傳送的資料封包每一個都必須經過SWITCH判斷決定要送往哪一台電腦，所以會有一些延遲，因此有時候電腦數少於五台，用HUB反而比SWITCH快。

**3.IP分享器(消費型簡易Router)：**  
藉由IP位址判斷封包應該往哪裡送。  
OSI模型層級 L3網路層

這個設備通常會有一個WAN port和1~4個不等的Lan port(其實是Switch)。WAN port一般是接ADSL modem(也就是小烏龜啦)，而Lan port則是接到PC電腦。這個設備主要的功能是NAT，也就是做IP分享的意思，他會將WAN port的真實IP(可以是固定IP或浮動IP)分享給LAN port的電腦使用。

**備註解釋一下NAT功能：**  
一般IP分享器LAN port裡的電腦室使用虛擬IP，也就是俗稱的假IP，這個網段通常是192.168.X.X，最常見的是192.168.0.X與192.168.1.X。這個IP網段是保留的網段，在實際網際網路並不能使用。NAT這個功能負責記錄網卡MAC位址與假IP的關係並做轉換。

舉例來說，假設使用者有兩台PC透過IP分享器上網  
A電腦IP為192.168.0.2，網路卡MAC位址是00046F12301A  
B電腦IP為192.168.0.50，網路卡MAC位址是000879215B09

當A電腦想要連網站yahoo.com.tw時，A電腦會送出資料給yahoo，而IP分享器會記錄192.168.0.2的MAC是00046F12301A，並把資料內來源IP 192.168.0.2改成WAN port的真實IP，然後送往YAHOO。Yahoo收到以後會依照真實IP位址回傳資料給IP分享器，IP分享器再依照資料內的MAC位址判別出這封包是要給A電腦的，於是把目的IP改成192.168.0.2送往A電腦。簡單的說，IP分享器是透過傳送封包內的MAC位址來分辨這個封包資料是要給哪一台電腦的，進而達到分享IP的功能。請注意，有些IP分享器會說自己是IP sharing router或者是寬頻Router，但這與實際的Router有一段差距。

**4.路由器(高階型Router)：**  
藉由IP位址判斷封包應該往哪裡送。  
OSI模型層級 L3網路層

路由器通常最少會有兩個介面，而這兩個介面分別區隔不同的IP網段。例如IP分享器有WAN和LAN兩種介面，區隔WAN的實際IP與LAN的虛擬IP網段，所以說IP分享器是Router其實並沒有錯誤。通常技術人員口中所說的router是高端路由器，而且兩個介面都是用在真實IP網段的設備。其實如果你對switch的原理有一點瞭解，那麼其實router有一點像switch，只不過Router是依照封包的目的IP來決定資料是要傳往哪一個介面的哪一個網域。

**5.Gateway閘道器:**  
將2 種使用不同傳輸協定的網路橋接在一起。  
OSI模型層級 L5會話層

**2) 何謂IP fragmentation?**

IP fragmentation (IP 分段)  
一個封包被拆分為兩個或多個封包。Symantec Protection Agent 支援 IP 分段，這是一種透過網路接收或傳送不完整封包的功能。

**3) 說明DHCP server功能**

DHCP伺服器主要功能就是用來配發IP，一來因為伺服器會自動配發IP而節省人力，二來不會發生IP重疊而互相衝突，三來可將IP回收循環使用(例：50個IP提供給錯開時段的70台腦使用)。  
DHCP能分配實體IP與虛擬IP，只要在DHCP Pool裡設定可分配的IP範圍，就會依需求自動分配IP。

**4) 說明IP、subnet mask**

IP位址（IP Address）  
在Internet上，一種給主機編址的方式。常見的IP位址，分為IPv4與IPv6兩大類。

A Subnet mask  
is a 32-bit number that masks an IP address, and divides the IP address into network address and host address. Subnet Mask is made by setting network bits to all "1"s and setting host bits to all "0"s.

**5) 說明 3-way handshaking**

三向交握 (Three-way handshake)  
TCP 通訊協定是一種 connection-oriented 協定，它在實際資料傳送前，會在來源端與目的端主機以三向交握 (three-way handshake) 的方式先建立連線，所有屬於相同訊息的 TCP 封包，就利用此連線傳送，此種作法有助於資料傳輸的正確性。以下是三向交握的流程說明：  
　  
(1) 當傳送端想要與接收端連線時，同時會啟用一個大於 1023 的通訊埠作為溝通的介面，並且送出一個要求連線的 SYN 封包，此封包內帶有起始序號 (例如 100)。  
(2) 接收端確認收到連線的 SYN 封包後，會回送一個 SYN+ACK 封包給傳送端，封包內則帶有回應號碼 (連線 SYN 封包的序號 + 1)，以及接收端的起始序號 (例如 2200)，等待傳送端的回應確認。  
(3) 當傳送端收到來自接收端的回應號碼，確認之前的連線要求封包已被收到，便會再傳送一個 ACK 封包給接收端，封包內會帶有回應號碼 (SYN+ACK 封包的序號 + 1)。  
　  
待接收端收到帶有正確回應號碼的 ACK 封包，此連線便正式建立!

本篇文章尚有頁面: [1](http://eeepage.info/interview-c/) [2](http://eeepage.info/interview-c/2/) [3](http://eeepage.info/interview-c/3/) [4](http://eeepage.info/interview-c/4/) [5](http://eeepage.info/interview-c/5/) [6](http://eeepage.info/interview-c/6/) 7 [8](http://eeepage.info/interview-c/8/) [9](http://eeepage.info/interview-c/9/) [10](http://eeepage.info/interview-c/10/)