Q : What ‘s the problem ?

```
#include <stdio.h>

int main()
{
   float f_var = 1.5F;
   double *d_ptr = (double *)&f_var;

   *d_ptr=2.5;                  // wrong way
   // *(float *)d_ptr=2.5;     // right way

   printf("%f\n",f_var);
   return 0;
}
```

結果：0.000000

如果你以 Visual C++6.0 去執行這支程式時，會得到以下的結果

爲什麼會得到這樣的結果呢？

Answer:

主要的問題在於程式中的這行程式：

```
*d_ptr=2.5;
```

程式中以 double 的指標 d_ptr，指向一個 float 的變數 f_var，接著透過 d_ptr 指標去更改變數 f_var 的數值。而實際的變數內容更改過程如下：

1) 2.5 常數值，是一個 double 的數值，佔 8 bytes
2) 將 2.5 的常數值，設定給 d_ptr 指標所指向的記憶體位置

是否注意到，d_ptr 指向的變數是 float 型態，佔 4bytes。現在要透過指標將 8 bytes 的常數值 2.5 間接設定給 4 bytes 的變數，這時會發生什麼事情呢？**這造成多出來的 4 bytes 會覆蓋到其它變數/暫存器的值，因而造成程式在執行時，讀取(執行)到非合法的數值(位址)**。透過 Visual C++ 6.0 來驗證此答案。

Note:

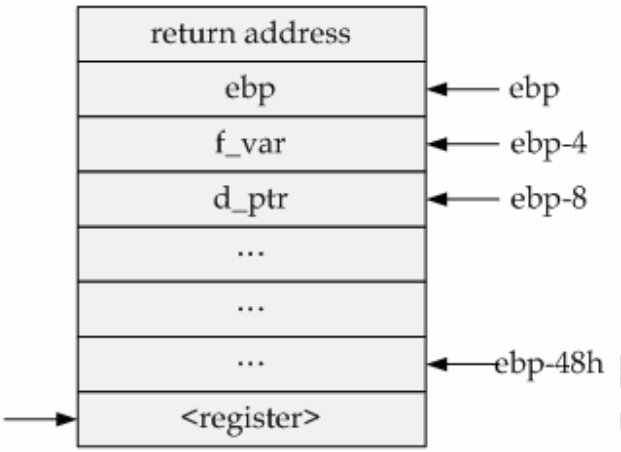
以下的說明會提到 Stack Frame，請自行閱讀作者寫得另一篇文章《Stack Frame 剖析》

http://home.kimo.com.tw/abc9250/LBEE_StackFrame.htm

首先來觀察以上 C 語言的程式所對應到的組合語言碼：

```
1:    #include <stdio.h>
2:
3:    int main()
4:    {
Stack Frame : Standard prolog code
004105C0   push      ebp               ;   Save ebp
004105C1   mov       ebp,esp           ;   Set stack frame pointer
004105C3   sub       esp,48h           ;   Allocate space for locals
004105C6   push      ebx               ;   Save register ebx
004105C7   push      esi               ;   Save register esi
004105C8   push      edi               ;   Save register edi
004105C9   lea       edi,[ebp-48h]     ;   Intialize space with 0x0CCCCCCC
004105CC   mov       ecx,12h
004105D1   mov       eax,0CCCCCCCCh    ;   0x0CCCCCCC: Uninitialized locals
004105D6   rep stos   dword ptr [edi]
5:      float f_var = 1.5F;
004105D8   mov       dword ptr [ebp-4],3FC00000h ;
6:      double *d_ptr = (double *)&f_var;
004105DF   lea       eax,[ebp-4]
004105E2   mov       dword ptr [ebp-8],eax
7:
8:      *d_ptr=2.5;
004105E5   mov       ecx,dword ptr [ebp-8]
004105E8   mov       dword ptr [ecx],0
004105EE   mov       dword ptr [ecx+4],40040000h
9:
10:     printf("%f\n",f_var);
004105F5   fld       dword ptr [ebp-4]
004105F8   sub       esp,8
004105FB   fstp      qword ptr [esp]
004105FE   push      offset string "%f" (0042601c)
00410603   call      printf (00410880)
00410608   add       esp,0Ch
11:     return 0;
0041060B   xor       eax,eax
12:   }
Stack Frame : Standard epilog code
0041060D   pop       edi               ;   Restore register edi
0041060E   pop       esi               ;   Restore register esi
0041060F   pop       ebx               ;   Restore register ebx
00410610   add       esp,48h
00410613   cmp       ebp,esp
00410615   call      __chkesp (00410840)    ; Check the esp
0041061A   mov       esp,ebp           ;   Restore stack pointer
0041061C   pop       ebp               ;   Restore register ebp
0041061D   ret                         ;   Return from function
```

一開始先建立Stack Frame來儲存Local Variable，而在程式中f_var與d_ptr變數在Stack Frame中的位置，如下圖所示：



當程式執行到底下這一行 C 語言的程式

```
8:      *d_ptr=2.5;
; 將 d_ptr 的值複製一份到暫存器 ecx，所以現在 ecx 的值爲 f_var 變數的位址，即爲 ebp-4
004105E5   mov       ecx,dword ptr [ebp-8]
; 將 0 值，複製一份至 ecx 所指的位址，即爲 f_var 變數
004105E8   mov       dword ptr [ecx],0
; 將 40040000h 值複製一份至 ecx+4 所指的位址，即爲 ebp 所在位址!!!
004105EE   mov       dword ptr [ecx+4],40040000h
```

此時已修改到舊有的 ebp 值。而當 main 程式執行結束之後，caller(呼叫 main 函式的函式)接收到不正確的 ebp 值，並透過 ebp 去做存取(Access)的動作時，「Access Violation (存取違規)」的錯誤訊息就產生出來。「0xC0000005 - Access Violation」這是一個十分常見的錯誤訊息，意思是指「存取的記憶體位址爲不合法」。