C Programming I
HW0606
Answer

作者: 吳文元 (jw910731)

日期: 2021/01/10

1 Bonus: What is the difference

兩支程式的執行結果完全相同，但其中記憶體位置的分佈卻有所不同我將兩隻程式皆加入以下程式片段在主程式的最後面

```
for (size_t i = 0; i < 9; i++){
    printf("%p\n", a[i]);
}
printf("%p\n", a);
```

1.1 程式 1

這是加入的程式之輸出:

```
0x7ffddbe784a0
0x7ffddbe784c4
0x7ffddbe784e8
0x7ffddbe7850c
0x7ffddbe78530
0x7ffddbe78554
0x7ffddbe78578
0x7ffddbe7859c
0x7ffddbe785c0
0x7ffddbe784a0
```

我們可以發現每行記憶體都相差 9*4 個 Byte，其中的 4 是`int_32`的大小。也就是說每一行的位置是完全相鄰、連續的，且完全由 Stack 管理 (也就是自動管理其記憶體的生命周期)。

1.2 程式 2

這是加入的程式之輸出:

```
0x556326a1f2a0
0x556326a1f2d0
0x556326a1f300
0x556326a1f330
0x556326a1f360
0x556326a1f390
0x556326a1f3c0
0x556326a1f3f0
```

```
0x556326a1f420
0x7ffc974f07b0
```

雖然可以發現每行記憶體相差 12*4 個 Byte，但這其實只是實作導致的巧合，此外陣列 a 的第一個維度與其他維度所在的位置是相差甚遠的！這是因為 a 這個陣列是在 Stack 上，而其元素指向的位置皆在 Heap 上，且由`malloc`與`free`函數手動管理這些元素指向的空間的生命周期！此外，這支程式有記憶體洩漏的隱憂存在，因其`malloc`的空間沒有被`free`掉

1.3 結論

這也就是說這兩支程式最大的差別是在記憶體管理的策略上，第 1 支程式使用自動的管理策略，將陣列放在 Stack 上面；第 2 支程式使用手動的管理策略，宣告了一個指標陣列在 Stack 上，且裡面的元素指向由`malloc`分配的 Heap 空間，但顯然管理不當，出現了記憶體洩漏的問題，幸好這支程式很快就執行結束了，並未造成問題。