C++ Tricks 2.2 I386平臺的內存佈局

從 farseerfc.wordpress.com 導入

2.2 I386平臺的內存佈局

衆所周知，I386是32位體系結構。因此對於絕大多數I386平臺的C++編譯器而言，sizeof(int)=sizeof(long)=sizeof(void*)=4。當然C++標準對此沒有任何保證，我們也不應該試圖編寫依賴於此的代碼。

32位指針的可尋址空間爲4GB。爲充分利用這麼大的尋址空間，也是爲了支持其它更先進的技術比如多任務技術或者動態鏈接庫技術，WinNT使用虛擬內存技術，給與每個應用程序全部4GB的內存空間。4GB的地址被一分爲二，前2GB供應用程序自己使用，後2GB由系統內核分配和管理。這2GB的內存地址，通常被劃分成3種內存區使用：

1 代碼及靜態數據區

由代碼加載器從動態鏈接庫鏡像(通常是exe或dll文件)加載，通常定位到鏡像文件中指定的基址開始的內存區。如果基址所在內存已被佔用，動態連接器會將代碼或數據重定向到其它可用地址。

在C++中，靜態數據包括：名字空間(namespace)和全局(global)對象、函數的static對象、類的static數據成員。這些靜態數據由編譯器分配地址(但可能被重定向)，由靜態連接器寫入代碼文件(通常是exe或dll)的靜態數據區段。所以標準說，這些靜態數據在編譯期就已經具有地址。

2 棧(Stack)

棧是最常用的動態數據存儲區，所有函數的non-static對象和函數參數都在程序運行期在棧上分配內存。在數據結構中，術語“棧(Stack)”意指先進後出(FILO, First In Last Out)，與“隊列(Queue)”所指的FIFO相對。相對於基於堆的對象分配技術，默認使用棧的對象分配有兩點優勢：

一、棧的FILO與人的思維方式相同

現實生活中有許多事例都使用FILO的方式，比如人們必須先提起話筒再撥打號碼，而後掛斷電話之後再放下話筒。使用FILO的棧，可以保證事物的銷燬順序以其誕生順序相反的順序進行，不會產生在掛斷電話之前就放下話筒的尷尬。

二、棧的分配管理僅需要兩個額外指針：棧頂(esp)和棧底(ebp)指針

從實現的技術層面而言，棧的管理比其它動態分配技術要簡單很多。I386平臺上的動態棧管理，僅需要棧頂和棧底兩個指針。這兩個指針的存儲顯然不能放置於棧中，置於靜態數據區又有損效率。I386平臺爲管理動態棧專門預留了兩個通用寄存器變量esp與ebp，分別代表棧頂(esp,Extended Stack Pointer)與棧底(Extended Bottom Pointer)指針。其中的extended代表它們是32位指針，以區分16位的sp和bp寄存

器。

棧是動態存儲區的特點，表明它的內存佔用將隨着程序的運行而變化。I386平臺上WinNT將應用程序的棧置於程序空間，向下增長。程序初始化時，由操作系統將esp指向系統分配的棧空間的頂部。當程序需要在棧上分配變量時，就將esp減去變量所需字節數，這被稱作“壓棧(Push)”；隨後又要銷燬變量時，就將esp加上變量所需字節數，這被稱作“彈出(Pop)”。esp與ebp兩者之間所夾的空間，就是當前函數正在使用的棧空間。由於棧向下增長，esp(棧頂)的值總是小於ebp(棧底)的值，新分配的變量地址總是小於舊變量的地址。

3 堆(Heap)和自由存儲區

棧中的變量對於分配與釋放的順序有特定要求，這在一定程度上限制了棧的適用範圍。面向對象(OO，Object Oriented)的程序設計思想也要求能自由地控制變量的分配與銷燬。由此，現代操作系統都提供了被稱作“堆(Heap)”的自由存儲區，以允許由程序員控制的對象創建和銷燬過程。C標準庫函數malloc和free則是對操作系統提供的堆操作的封裝。C++提供的自由存儲區運算符new和delete則通常是malloc和free的又一層封裝。

操作系統經由malloc和free控制對堆的訪問。堆的存儲管理技術各不相同，簡單的使用雙鏈表管理，複雜的可以比擬一個完整的文件系統。

由於額外的管理需求，使用系統提供的通用分配器在堆上分配和銷燬變量的代價，無論從空間角度還是效率角度而言，都比在棧上分配對象要高昂很多。對於sizeof上百的大型對象，這樣的高昂代價還是可以接受的，但是對於sizeof只有個位數的小對象，這樣的代價通常是一個數量級的差距。正因爲這個原因，STL不使用new和delete，轉而使用分配子(alllocor)分配對象。