**Lab 4**

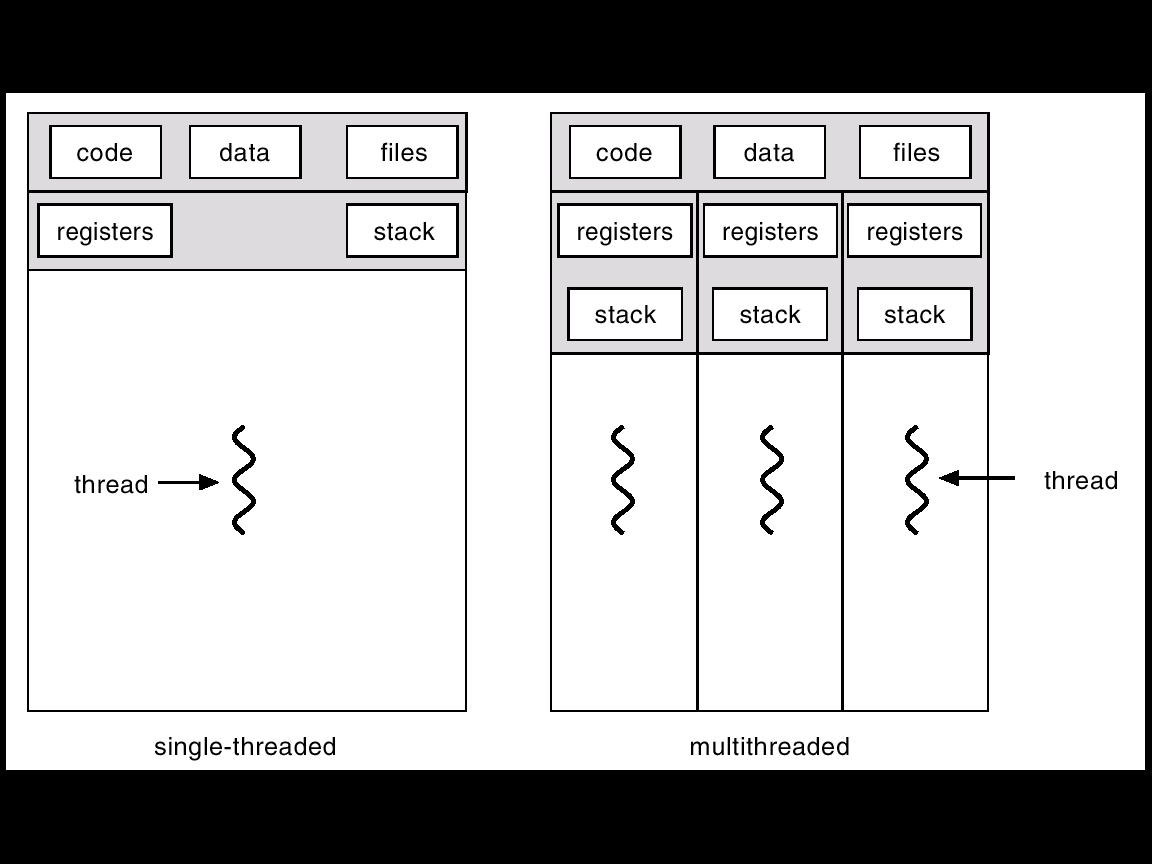
**multithread**

1. **實驗目的**

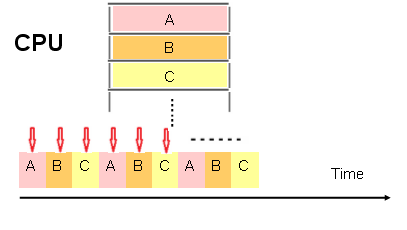
在上一次的實驗，我們成功地透過socket接收來自server的訊息，達到最簡易版瀏覽器的效果。那麼如果我們想要同時連上多個網站，並隨意執行任何一個socket的send 及 receive 函式，要如何實現呢？Visual C++ 本身提供了一個BackgroundWorker元件，這個元件本身即是一個以thread(執行緒)為基底所設計的元件，在本次的實驗中，我們實際來學習如何以BackgroundWorker來做為網路傳輸的背景設計。

1. **架構說明：**

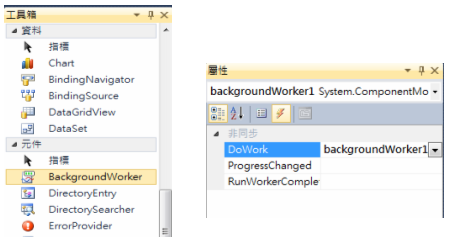
Process 中的一段程式碼執行軌跡稱為 thread，是電腦中最小的執行單位。



由於如果主程式內有迴圈的話，程式執行後會當住，且螢幕也不會顯示，因此，我們必須透過執行緒來解決此問題。利用time slicing 的概念，將時間軸切成好幾塊,每一小塊區間都執行不同的thread。



1. **實際操作：**
2. 利用visual studio 裡面的元件Backgroundworker，並且把該thread要執行的事情寫在DoWork函數



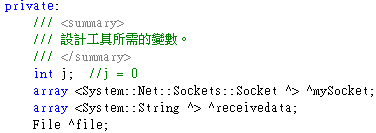
1. 利用上次實驗學到的來建立連線並取得html



1. 呼叫RunWorkerAsync()函式，觸發DoWork事件



1. A thread無法更改B thread的物件，可以利用global變數當作溝通的橋樑



1. **動態產生BackgroundWorker物件：**

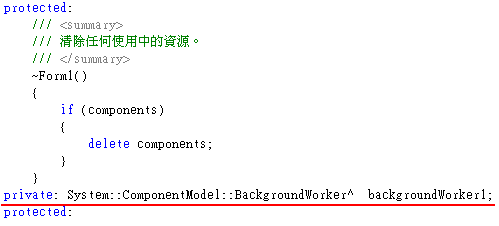
快速複習動態一維陣列的產生方式

int \*a;

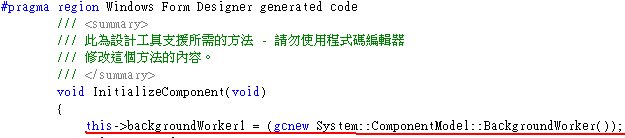
a = new int[5];

當我們從工具箱產生一靜態的BackgroundWorker物件並決定好其DoWork事件的event handler後，程式碼便自行處理上述動作

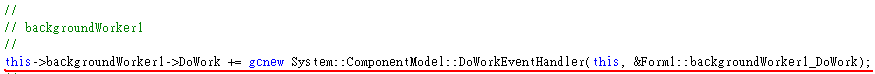
* 1. **宣告一個名為backgroundWorker1的BackgroundWorker pointer**



* 1. **給予空間，呼叫BackgroundWorker的建構子**



* 1. **使函式backgroundWorker1\_DoWork與backgroundWorker1的DoWorkEventHandler作連結**



有了以上的觀念，我們便可以取代從工具箱產生的作法，在程式執行的狀態下，依據不同的情況，產生動態的BackgroundWorker一維陣列。