EMBARCADERO
TECHNOLOGIES®

2011 CB/DELPHI 技術講座:
Multithread 之應用與實作:

蕭沖
仲鼎科技軟體部經理

EMBARCADERO TECHNOLOGIES®

前言

感謝捷康科技Qcom用心開辦講座

感謝KTOP的會員支持

感謝Embarcadero對CB / Delphi的繼續研發

EMBARCADERO TECHNOLOGIES

講座大綱

Multi-Thread.xmind

- ✓Thread的定義
- ✓Multithread的使用時機
- ✓Multithread與Timer的比較
- ✓Multithread的分類
- ✓同步問題
- ✓VCL建立Thread的方式
- ✓VCL的同步問題
- ✓VCL的Multithread解決策略

EMBARCADERO TECHNOLOGIES®

Thread的定義

程式設計師每天寫程式，程式到底是什麼?

EMBARCADERO TECHNOLOGIES®

Thread的定義

Alan

Turing

Thread的定義

Turing 的原始研究: 問題是否能用計算機來解決?

(一)定義一種抽象的計算機;

(二)證明萬用計算機 (universal machine) 的存在性;

(三)證明存在有任何計算機都不能解決的問題。

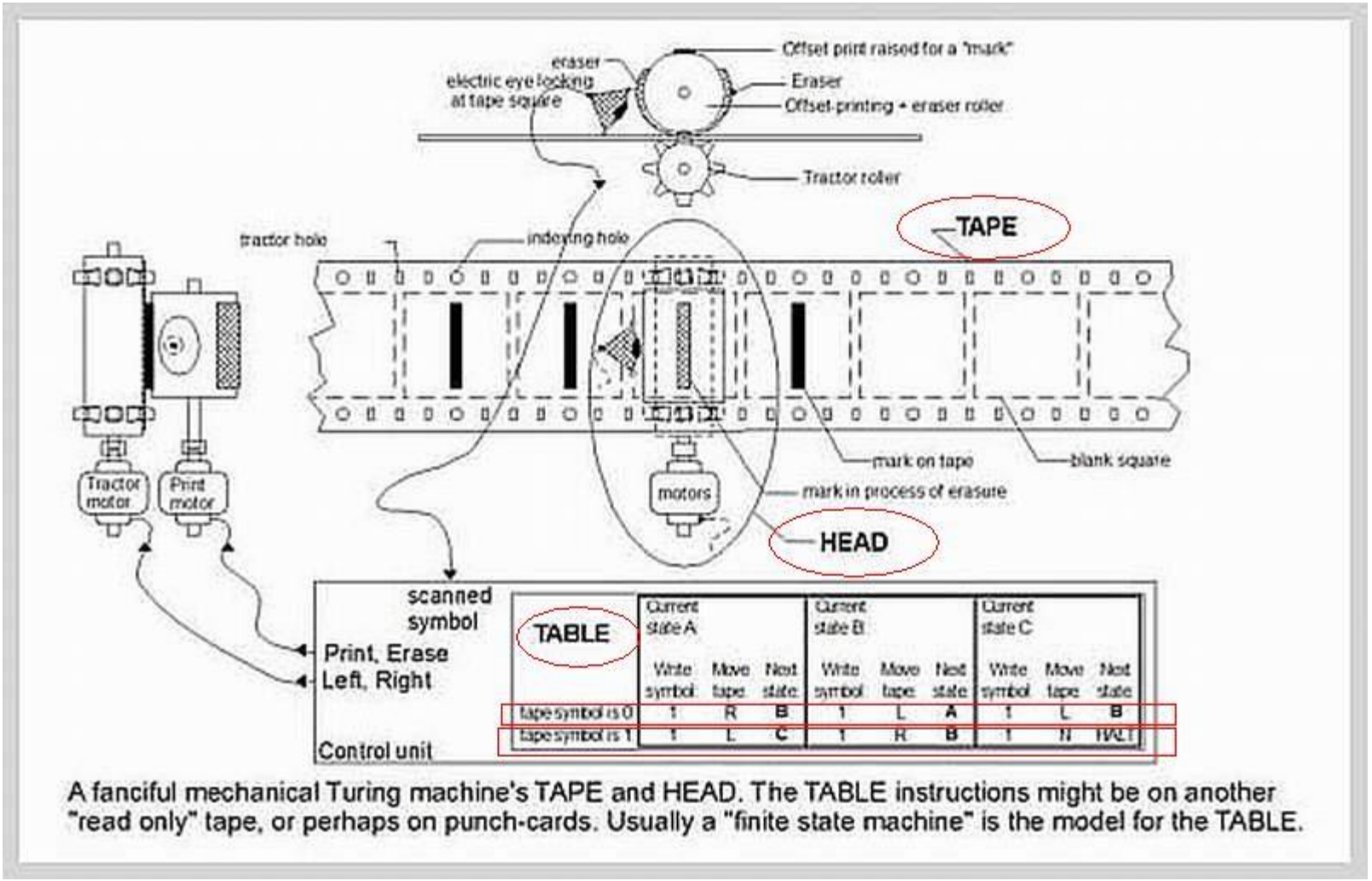
Church-Turing Thesis :
定義出什麼叫演算法 (Algorithm)—
能被 Turing Machine 計算出來的運算就是一種演算法

Thread的定義

Turing Machine (圖靈機):

- 一條無限長的紙帶TAPE。
- 一個讀寫頭HEAD。該讀寫頭可以在紙帶上左右移動，它能讀出當前所指的格子上的符號，並能改變當前格子上的符號。
- 一套控制規則TABLE。它根據當前機器所處的狀態以及當前讀寫頭所指的格子上的符號來確定讀寫頭下一步的動作，並改變狀態暫存器的值，令機器進入一個新的狀態。
- 一個**狀態暫存器**。它用來保存杜林機當前所處的狀態。圖靈機的所有可能狀態的數目是有限的，並且有一個特殊的狀態，稱為*停機狀態*。

Thread的定義　Turing Machine (圖靈機):



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

EMBARCADERO TECHNOLOGIES®

Thread的定義　Turing Machine (圖靈機):

運作原理(每個步驟):

輸入部份:

①當下狀態

②當下紙帶位置上的符號值

動作部份:

①在當下的位置寫入值

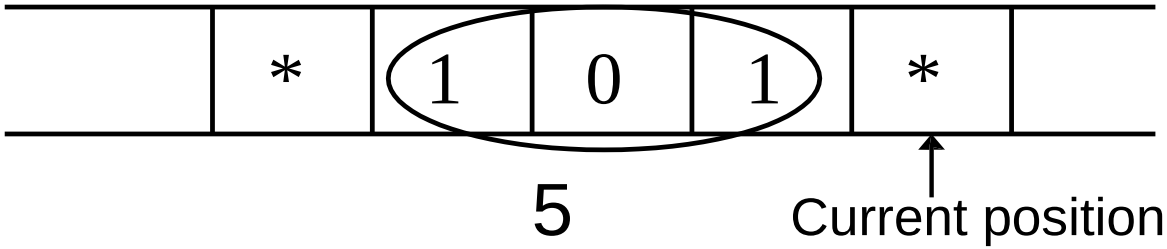
②往左或右移動讀寫頭至下個位置

③改變狀態

Thread的定義　圖靈機 實例解說

紙帶上有1，0，＊ 三種符號

我們把1與0當作二進制的數值



接著我們要實作把紙帶上的數值加1的圖靈機語言

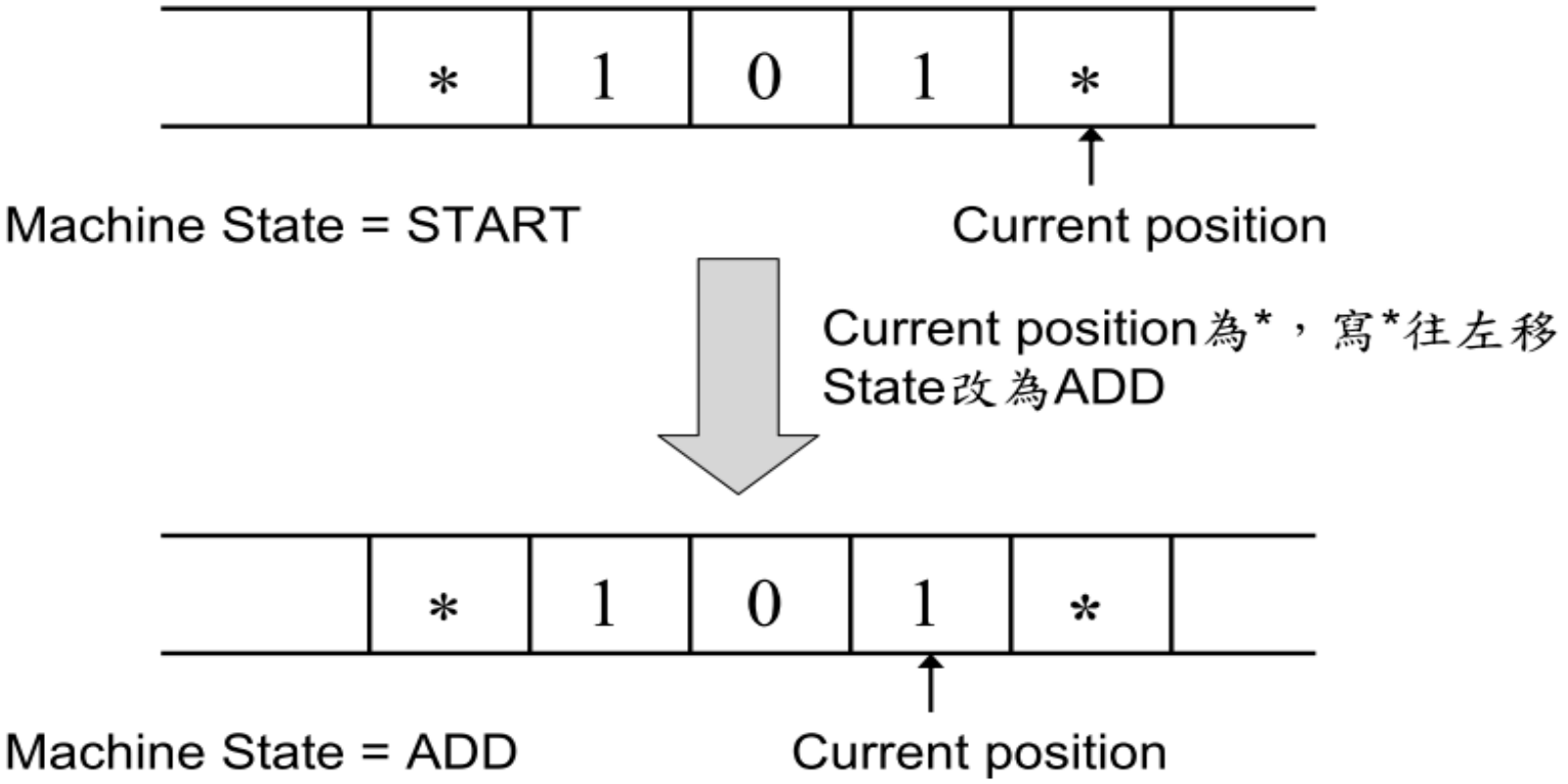
Thread的定義　圖靈機 實例解說

TABLE ： 狀態表如下:

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

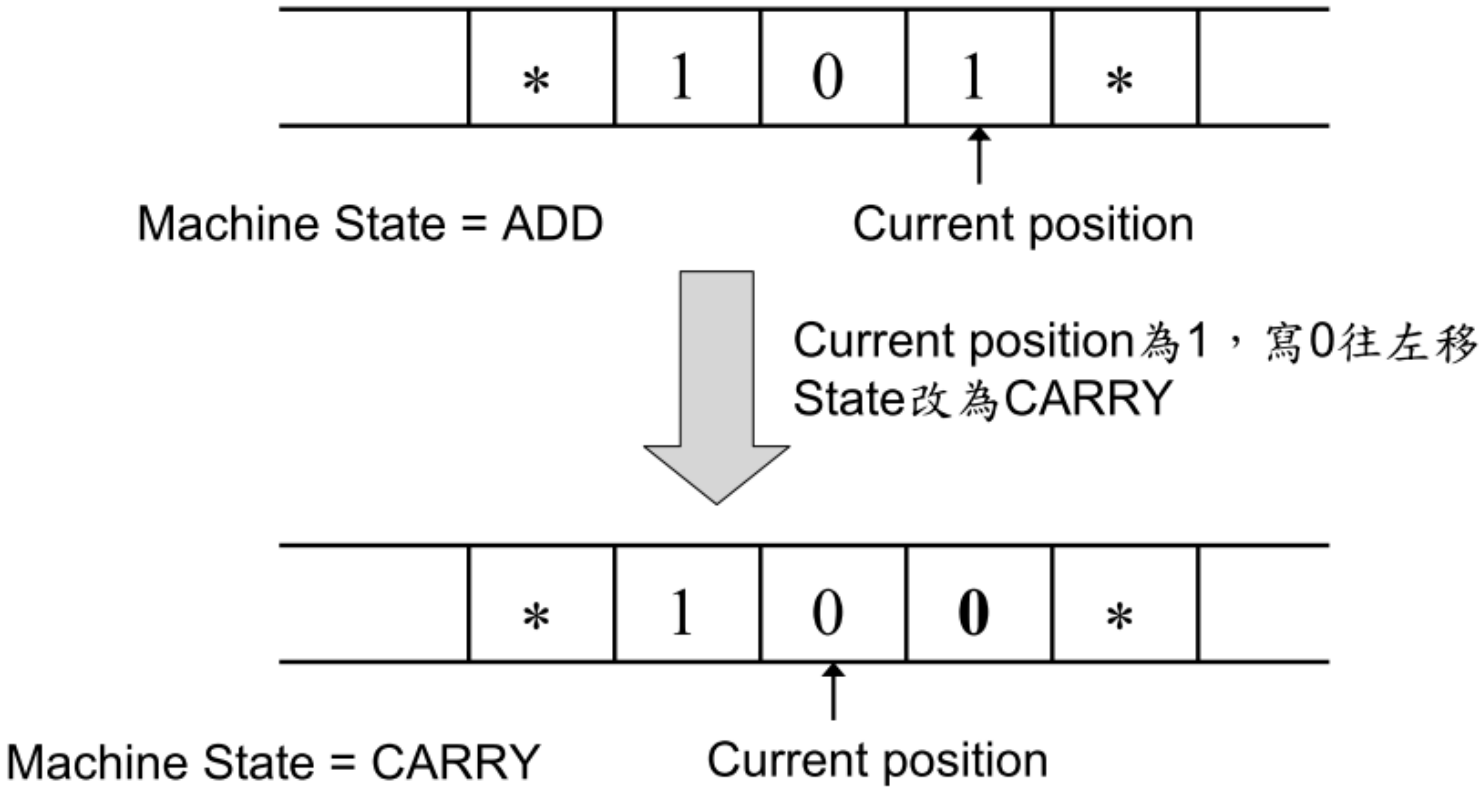
Thread的定義　圖靈機 實例解說

第一步驟 ：



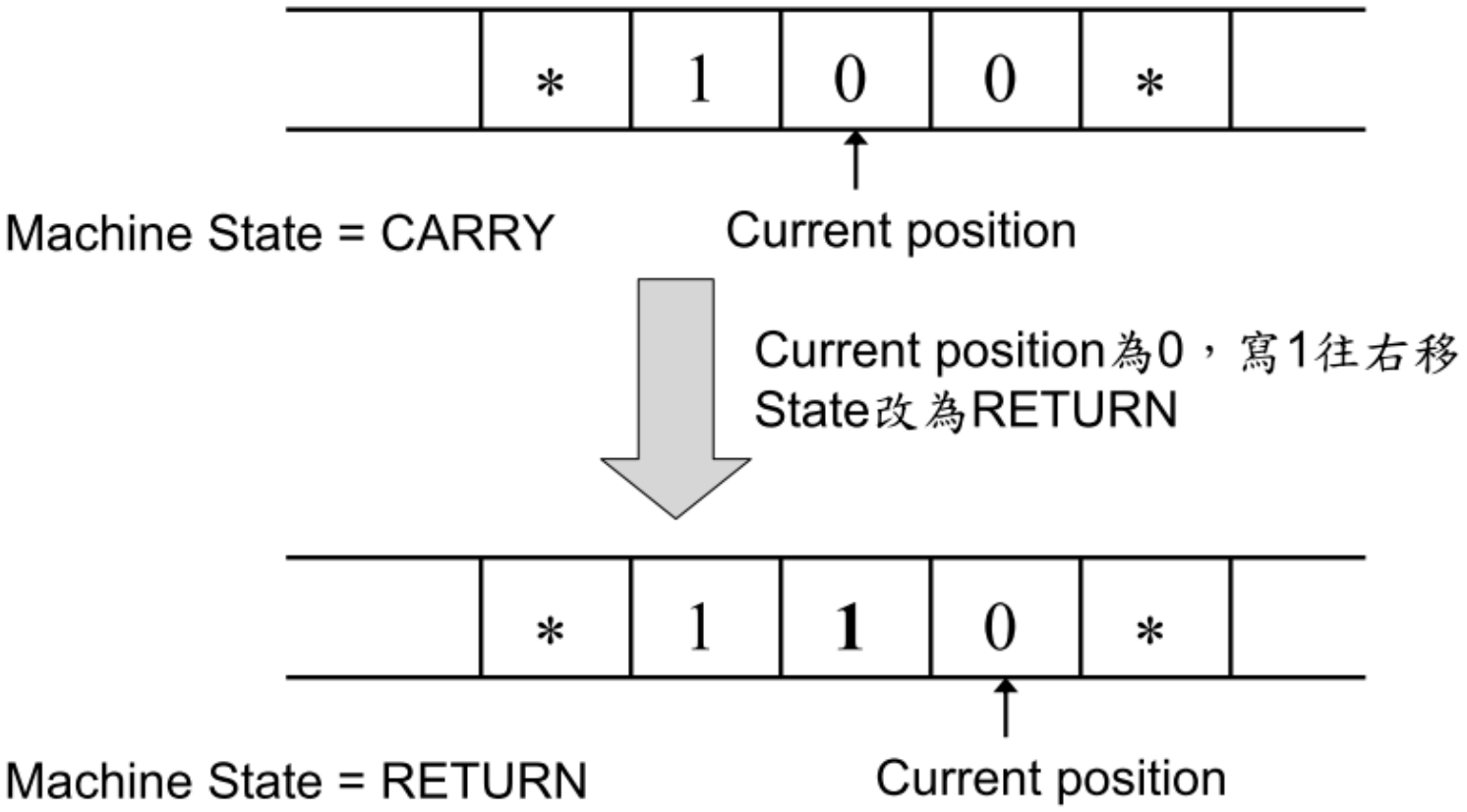
Thread的定義　圖靈機 實例解說

第二步驟 ：



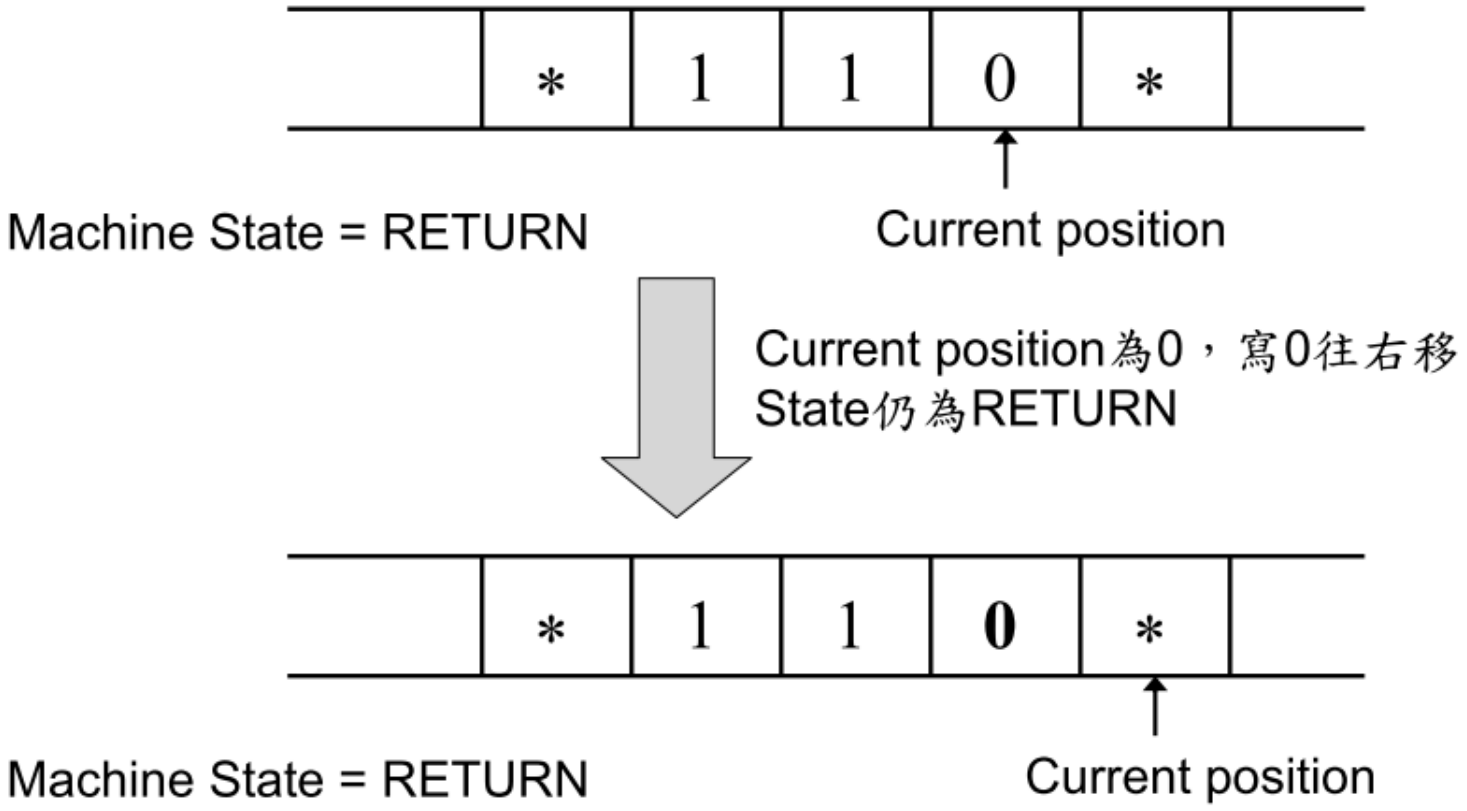
Thread的定義　圖靈機 實例解說

第三步驟 ：



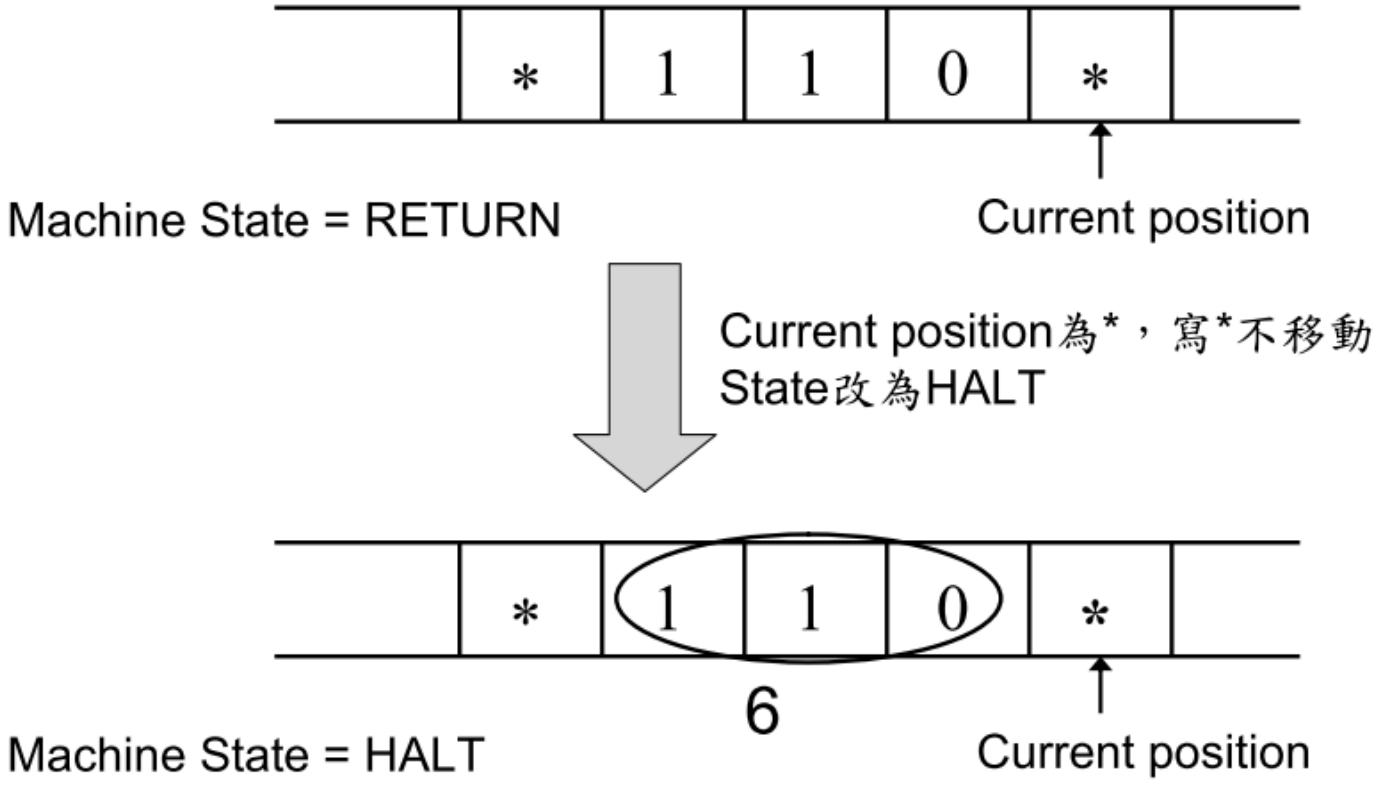
Thread的定義　圖靈機 實例解說

第四步驟 ：



Thread的定義　圖靈機 實例解說

最後步驟 ：



EX1

EMBARCADERO TECHNOLOGIES®

Thread的定義

程式是一種演算法，為了計算出某種結果。它就是Turing Machine的TABLE部份。

而Thread就是其中的「步驟」：即靠TABLE 的指示將HEAD左右移動，並讀與寫的一連串「循序」的讀寫「動作」。

Thread的定義

Thread 與 Process 的不同 (以 WIN 32 API的角度來看)

在硬碟裡的程式叫程式，程式被「載入並配置」在恰當的記憶體中後，那一整段的程式碼叫Process。

Process 猶如肉體 Thread 猶如靈魂。缺一不可!

CreateProcess這個API事實上不僅造出Process，其實也安排好Thread了。

CreateThread這個API則是一定要指出”Process”(ThreadFunc)的所在。

```
hThread = CreateThread(
    NULL,                   // no security attributes
    0,                      // use default stack size
    ThreadFunc,             // thread function
    &dwThrdParam,           // argument to thread function
    0,                      // use default creation flags
    &dwThreadId);           // returns the thread identifier
```

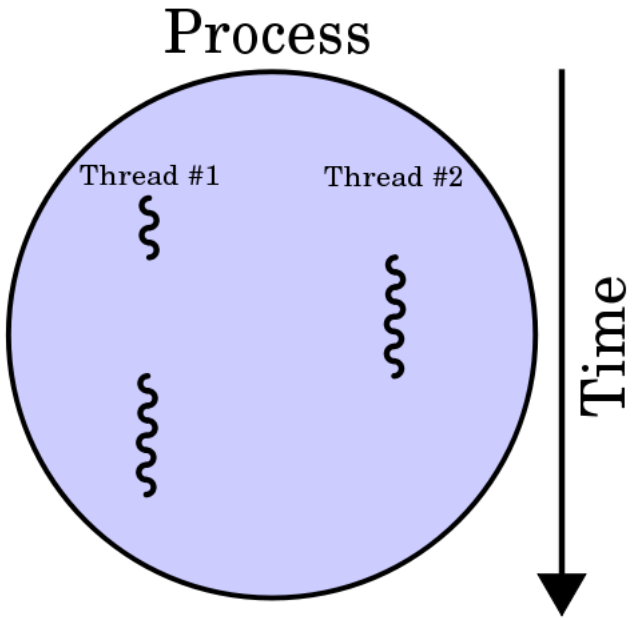
EX2

Thread的定義

一般來說一個CPU只有一條「原」Thread 的資源。從開機到載入OS…

DOS時代的應用程式就一條Thread用到底。

Windows時代，「原」Thread被OS切成數條，應用程式本身也可分配到不只一條Thread來用。



EMBARCADERO TECHNOLOGIES®

Multithread的使用時機

生活上的實例:　效能與人性

麥當勞

吉野家

EMBARCADERO TECHNOLOGIES®

Multithread的使用時機

為什麼不一條緒直直下去?

- 因為有週邊設備IO較慢且耗時
- 使用者是否可以忍受一段時間(數秒)沒有回應?(10s)
- 程序的時效性(即時性)的考量，如複合多通訊實作

Multithread的使用時機

程式上的舉例:

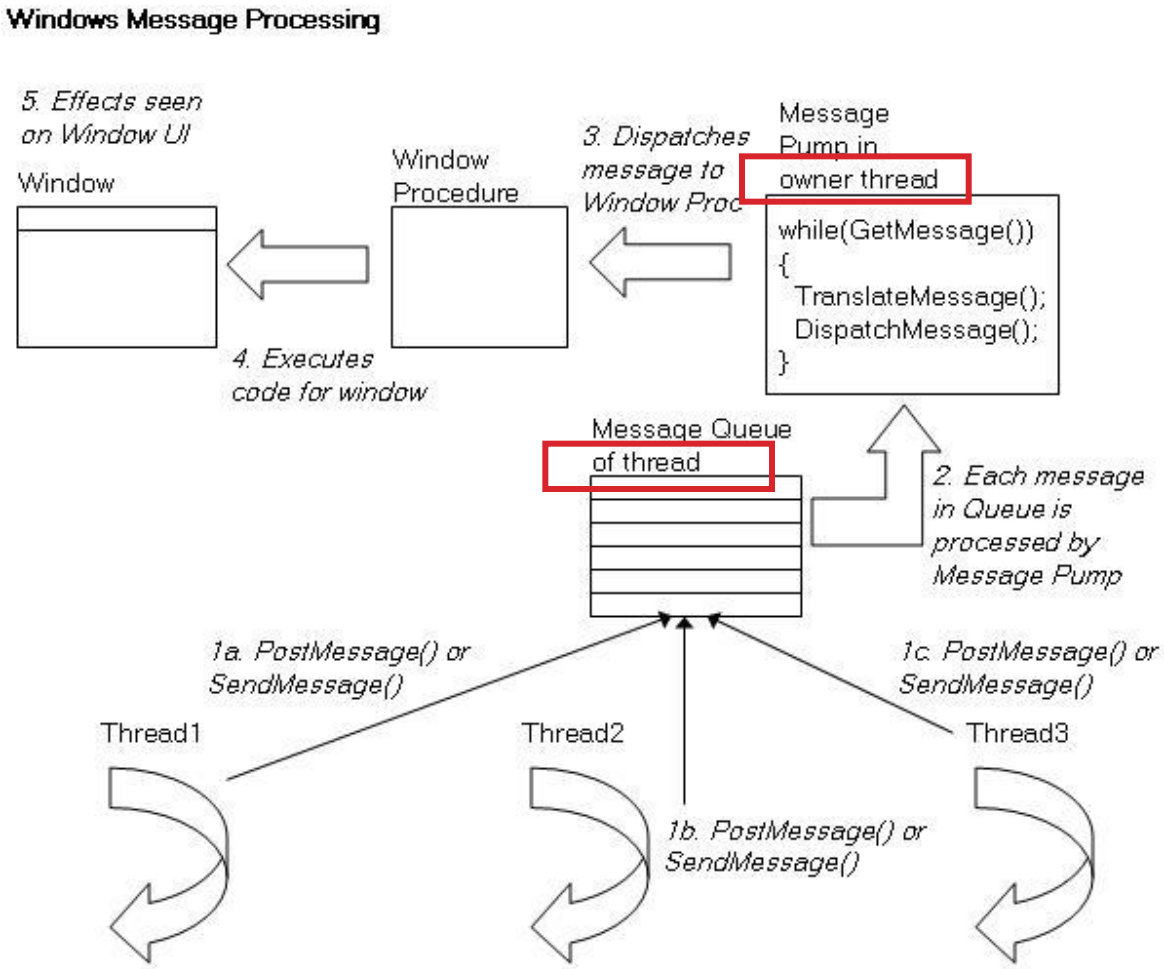
- 通訊測式程式: 連接好多設備同測
- Web Spider 程式
- 大量的資料庫搜尋

Multithread 的 缺點 :

Context Switch耗資源，若是沒有IO的考量與使用者的問題，沒必要用!

Multithread與Timer的比較

佇列是專屬於某個Thread的，但不是每個Thread都會有佇列。



Multithread與Timer的比較

- Timer仍是在同一個主thread下。
- Timer是一種訊息。
- The WM_TIMER message is a low-priority message。
- 雖然訊息會佇列，但在佇列區裡僅會有一個，多的無效。
- Timer的觸發時間是不可預期的。

EX3

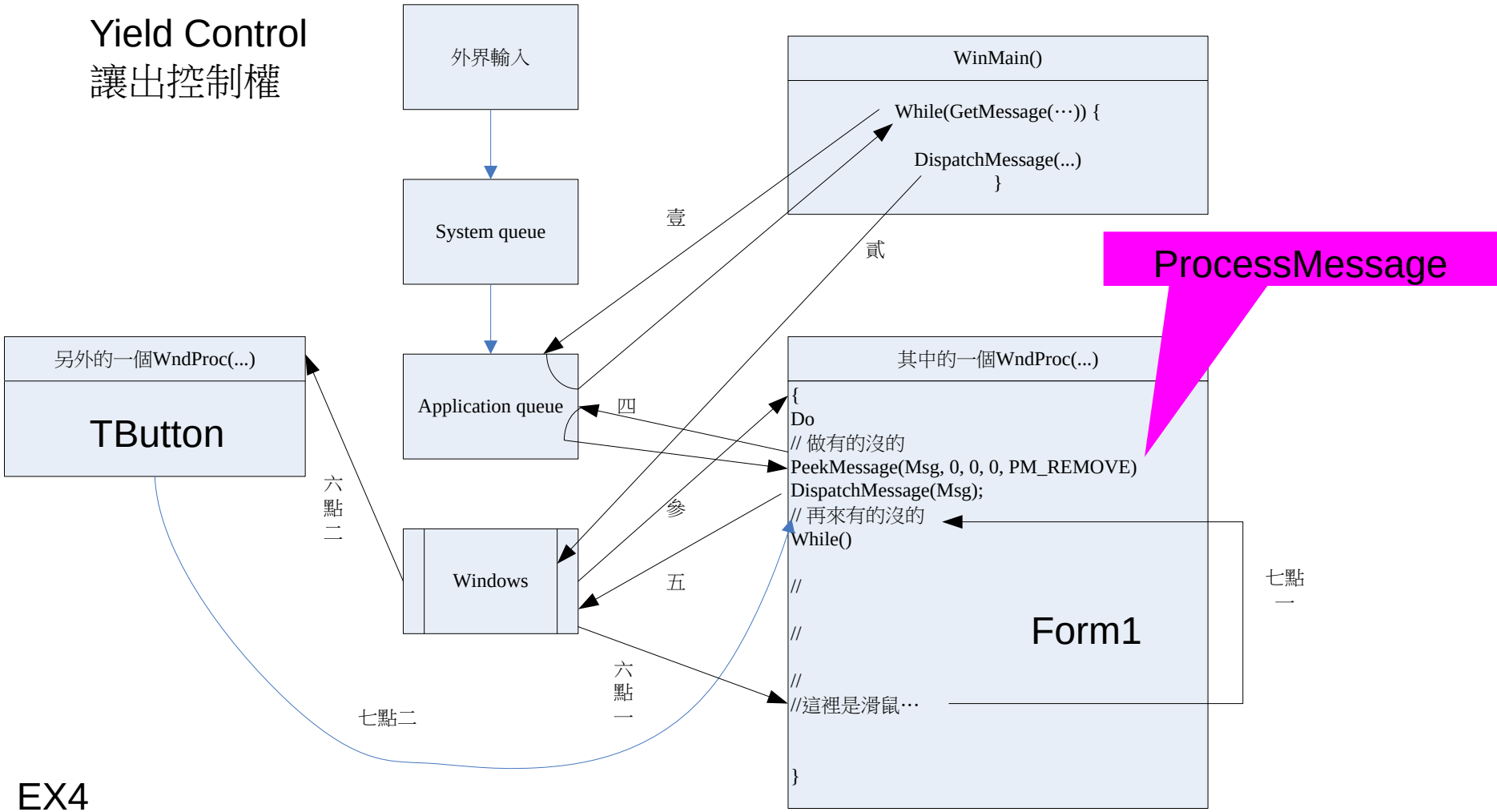
Multithread與Timer的比較

加入Application->ProcessMessage()來改善!!??

```
function TApplication.ProcessMessage(var Msg: TMsg): Boolean;
……..
begin
  Result := False;
  if PeekMessage(Msg, 0, 0, 0, PM_REMOVE) then
  begin
    Result := True;
    if Msg.Message <> WM_QUIT then
    begin
……
      if not IsHintMsg(Msg) and not Handled and not IsMDIMsg(Msg) and
        not IsKeyMsg(Msg) and not IsDlgMsg(Msg) then
      begin
        TranslateMessage(Msg);
        DispatchMessage(Msg);
      end;
    end
    else
      FTerminate := True;
  end;
end;
```

Multithread與Timer的比較

Yield Control
讓出控制權



EX4

Multithread與Timer的比較

結論:

- Timer不能夠完善的多工。
- 即使加上Application->ProcessMessage()會改善，但重入的問題可能帶來更大的隱憂。
- 適合用於不重要的多工處理，即lost幾次沒關係

Multithread的分類

定義: 除主thread以外，尚有至少一個子thread。

Multithread的分類

同程式區塊(thread process)，被建立n個thread

不同程式區塊，各自建立數個thread

混合型

Multithread的分類

以虛擬速食餐廳點餐建立一個Multithread極小程式

EX6

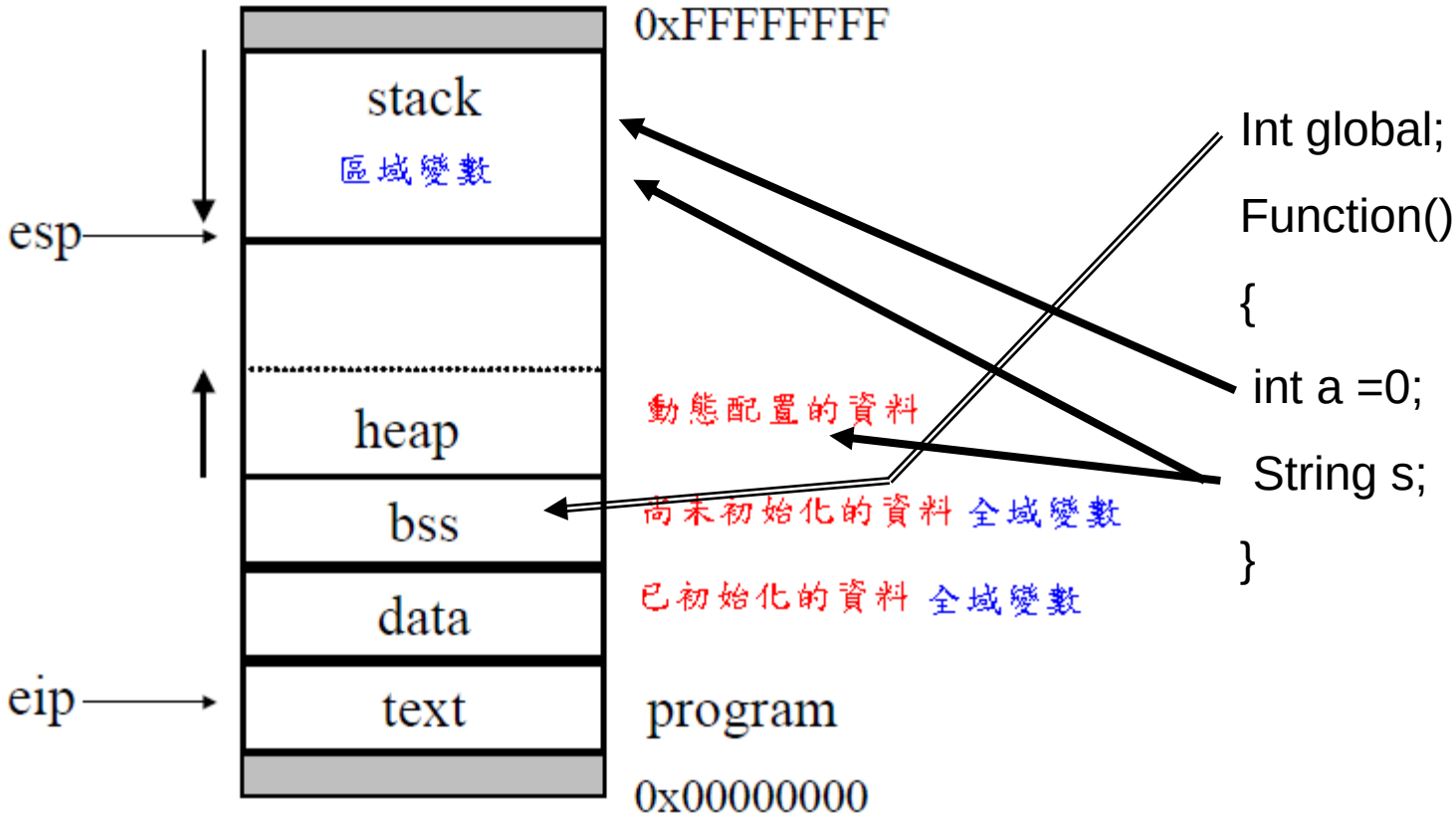
EMBARCADERO TECHNOLOGIES®

同步問題(一)　這是什麼問題?

共用一台飲料機…

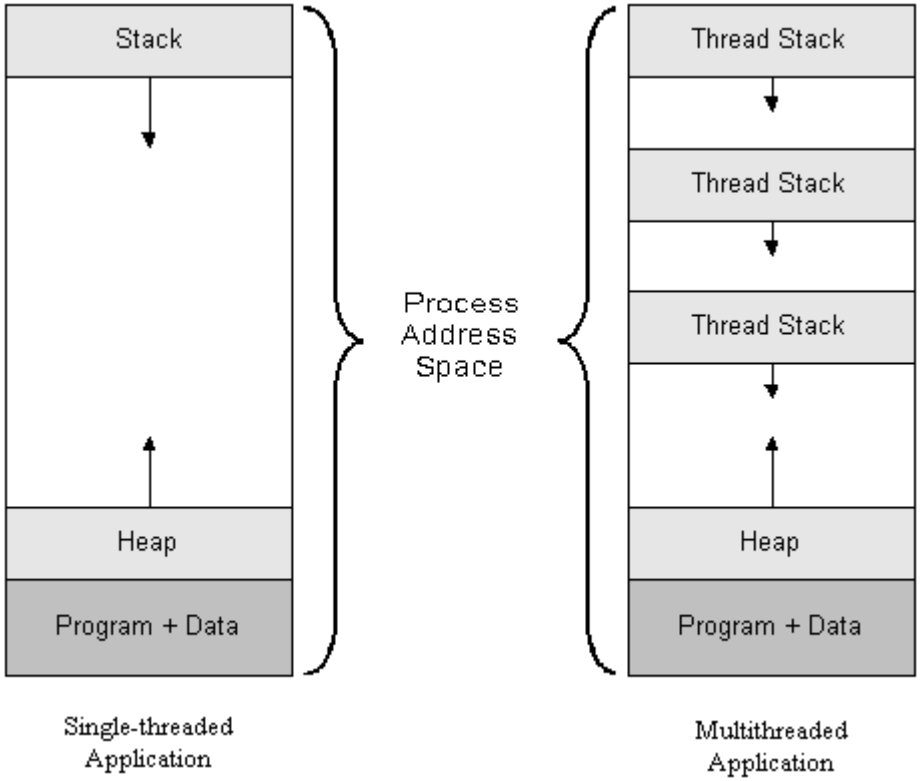
同步問題(一)　同時間存取同一物件

程式的記憶體配置



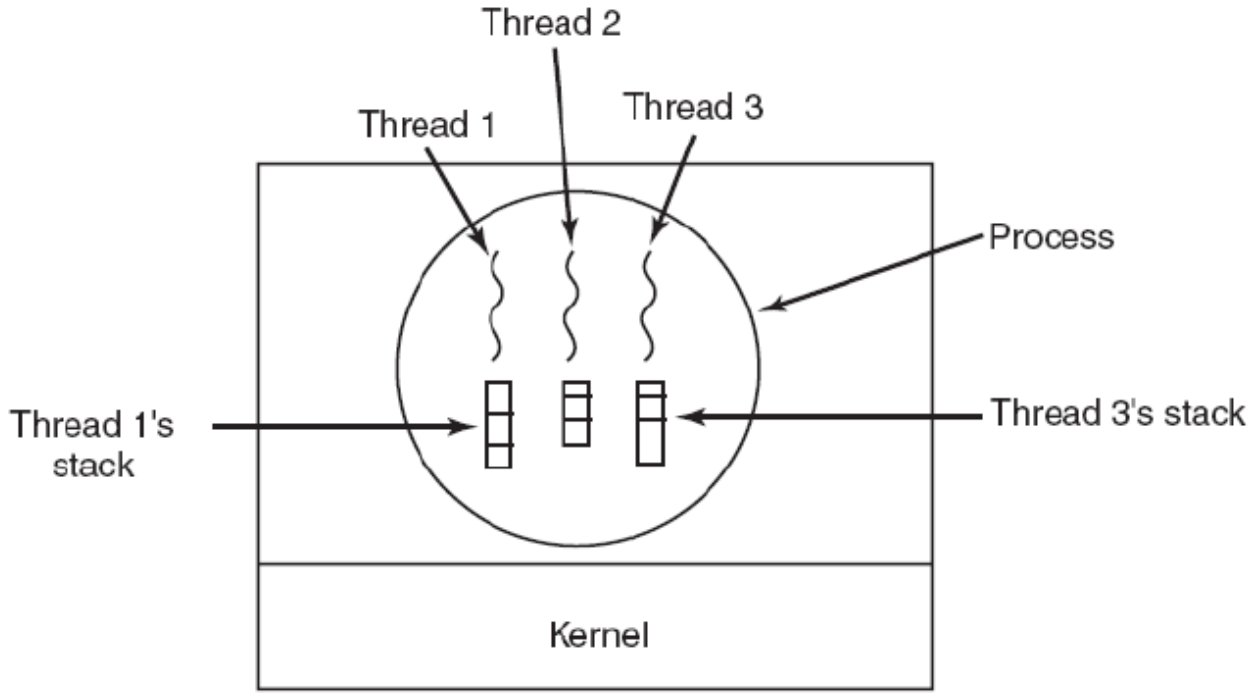
同步問題(一)　同時間存取同一物件

每個Thread所分到的專屬記憶體---Stack區



同步問題(一)　同時間存取同一物件

每個Thread所分到的專屬記憶體---Stack區



EMBARCADERO TECHNOLOGIES®

同步問題(一)　同時間存取同一物件

Thread-Safe 的「函式」驗證：(非方法)

1. 是否有使用全域的變數?
2. 是否有對全域變數做寫入的動作?
3. 變數是否只是單純的型別?　（不嚴僅的）
4. 該變數是否僅被參考一次?　（不嚴僅的）
5. 函式內所用的函式或函式庫也要合於1-4點。(strtok..)

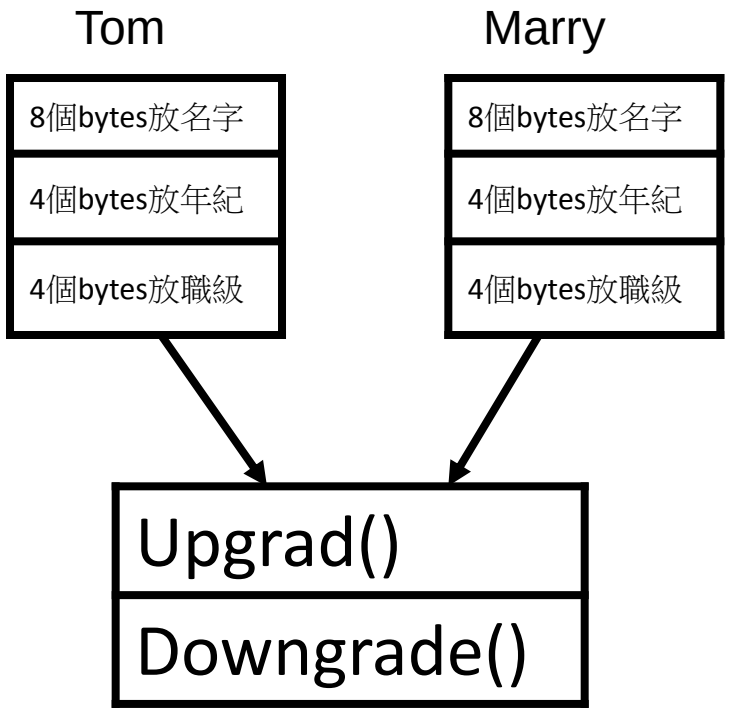
EX6,10

同步問題(一)　同時間存取同一物件

類別所創的實例(Instance)/物件(Object)的記憶體配置:

```
Class Employ
{
   char Name[8];
   int age;
   int grade;
   public:
   upgrade();
   downgrade();
}
```

```
Main()
{
  Employ tom;
  Employ marry;
}
```



方法是共用的!!!

EX7

同步問題(一)　同時間存取同一物件

Thread-Safe 的「類別」驗證 ：

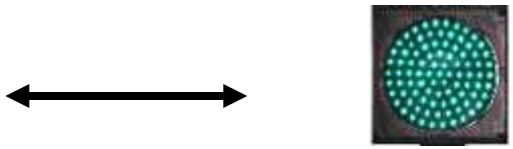
所有該類別的方法都是安全的時，則稱此類別是Thread-Safe的!

使用類別方法的額外考量:

1. 方法與函式的不同 ：(方法有資料成員)。
2. 檢查資料成員是否共用(即是否僅有一個instance)?

同步問題(二)　解決策略

掛上一個號誌，若使用中則紅，沒人用時就綠



同步問題(二)　解決策略

- Critical Section – 燈
- EnterCriticalSection -- 變紅
- LeaveCriticalSection -- 變綠

同步問題(三)　解決策略

- Mutex – 燈
- OpenMutex -- 試著開看看…
- 在Wait Funtions(32API)中放入，可得權。
- ReleaseMutex -- 變綠

同步問題(四)　解決策略

- Event Object -- 燈
- SetEvent -- 開
- ResetEvent -- 關
- WaitForSingleObject -- 等待直到開或逾時

EMBARCADERO TECHNOLOGIES®

同步問題(五)　解決策略

- 所有同步方案的比較
- 漏了一個叫Semaphore
- 介紹Kernel Object
- Critical Section，Event，Mutex，Semaphore
- 各種方式的特色—速度與範圍

VCL 建立Thread的方式

- TThread
- Resume / Suspend
- Kill Thread

TThread的使用注意事項

- 不能動態新增視窗元件

—ShowMessage (不可)

—Create Form (不可)

—MessageBox (此為Win32 API，可以)

TThread的使用注意事項

- CB XE與CB6 Tthread的比較

—TThread.Queue (像自己寫自訂msg然後PostMessage)

—TThread.RemoveQueuedEvents

—TThread.Sleep (suspend一段時間)

—TThread.Yield (交出控制權)

—TThread.SpinWait (不交出控制權，但等候)

VCL的同步問題

- 再論類別的Thread-Safe問題：

➢ 普通級的Thread-Safe類別：

此類別若個別建立使用則安全。

➢ 無敵級的Thread-Safe類別：

此類別的instance可完全被共用(僅一份)也安全--- 意味著有對資料成員做Multiplexing。(有解決資料成員同步問題)

VCL的同步問題

- 論VCL 類別的Thread-Safe問題 :

  官方說法:

  --BDE，ADO是thread-safe的。經我驗證是”普通級”的

  --但 data-aware controls 是不安全的。

  --Graphics objects are thread-safe。可使用它內建的lock方法。目前看來像是”無敵級”的。

  --TList是不安全的，但TThreadList則可使用lock的方式變成”無敵級”的。

VCL的同步問題

- 論VCL 類別的Thread-Safe問題 :

  官方沒提到的是否就一定是不安全?

  Tmemo

  Timage

  自己source debug看看…

EX13,13_2

VCL的Multithread解決策略(一)

- Synchronize的方法
- 原理: 把要做的事放入一個全域的list裡，然後自己進入等待狀態，直到main thread有空閒處理掉那個待辦事項後才返回。
- Synchronize.txt
- 只要是你懷疑有安全疑慮的，通通放入就安全。
- 效能非常的差!

VCL的Multithread解決策略(二)

SyncObjs單位裡的物件 --

- TEvent
- TCriticalSection
- TCriticalSection

VCL的Multithread解決策略(三)

SyncObjs單位裡的物件 --

- TThreadList

VCL的Multithread解決策略(三)

SyncObjs單位裡的物件 - XE版本才有的

- TMutex
- TInterlocked

VCL的Multithread解決策略(四)

補充題:

很有用但又神密的SendMessage 這個API

至同一thread時的行為 :

至不同thread時的行為 :

Send時本身不全然就是卡住，其實也可以接受外來的SendMessage.....

結束 – 謝謝大家!

本次Slide的內容參考:

Wikipedia

Google 圖庫

高雄大學—嚴力行教授—圖靈機例子

Msdn

CB help file

Delphi 源碼

感謝以上的資料來源!