第12章執行緒

資訊科技系
林偉川

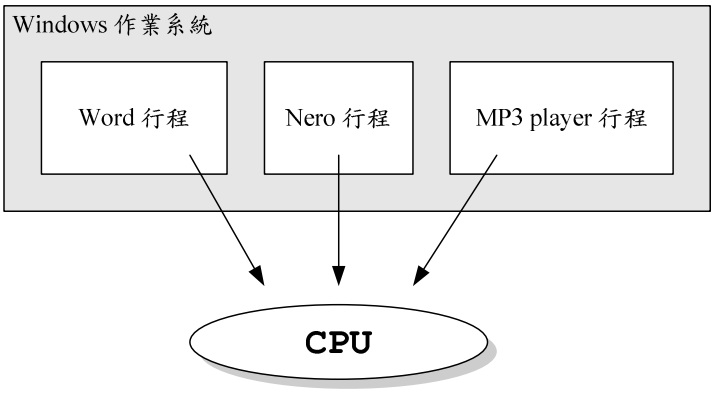
執行緒

- 了解行程和執行緒的不同
- 能使用Thread類別和Runnable介面建立執行緒
- 了解使用Runnable介面建立執行緒的優點
- 能分辨執行緒的5種狀態
- 知道在什麼情形下，執行緒將進入等待狀態
- 了解執行緒優先權值的意義
- 了解何謂執行緒同步
- 定義synchronized修飾方法和synchronized敘述
- 了解wait()和notify()方法如何使用

2

12-1-1 行程

多個行程使用單一CPU



3

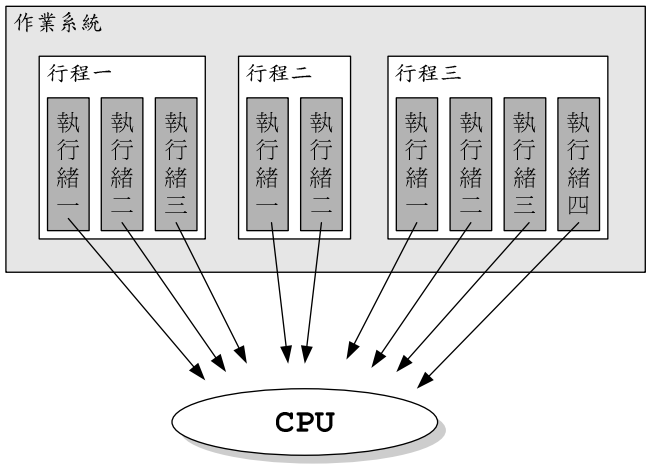
12-1-1 行程

- 不同的行程所佔有的記憶體資源不同，各自獨立互不干擾。
- 大部份的個人電腦只有一個CPU，所有行程都必須透過系統取得CPU的使用權。
- 每個行程輪流使用CPU的情形就像是每個行程都同時執行一樣。

4

12-1-2 執行緒

- 多行程多執行緒使用單一CPU



5

12-1-1 行程

- 執行緒是行程中的子程式，這些子程式共享行程內的資源，同時也可以分配到CPU的使用權。
- main()其實就是Java應用程式的預設執行緒(或稱主執行緒，Main Thread)。
- 多執行緒程式必須靠主執行緒去啟動其它執行緒的進行。
- Java的執行緒都必須是java.lang.Thread類別的物件。

6

- Thread的建構子及啟動、暫停方法

Thread之建構子及方法	說明
Thread(String name)	建立執行緒，同時設定執行緒的名稱為name。
final String **getName**()	取得執行緒的名稱。
static void **sleep**(long ms) throws InterruptedException	讓執行緒暫停ms毫秒的時間(若暫停的期間執行緒被中斷，則會丟出InterruptedException例外)。
void **run**()	執行緒啟動時所執行的方法。
void **start**()	讓執行緒啟動的方法。

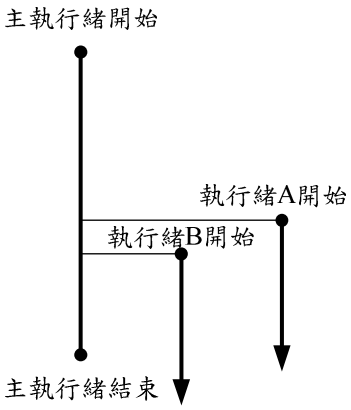
7

- 欲啟動一個執行緒時，必須呼叫它的start()方法，而不是直接呼叫它的run()方法。
- 若直接呼叫Thread類別物件的run()方法，並不會啟動執行緒，只是一般方法的呼叫。
- 執行緒結束之後，執行緒物件還存在，但是不可以重新啟動，除非執行System.exit(0) ，才能結束。

8

呼叫start()方法後執行緒才開始執行

```
public static void main(String [] args) {
        EX12_1 ta = new EX12_1("執行緒A");
        EX12_1 tb = new EX12_1("執行緒B～");
        ta.start();
        tb.start();
    System.out.println("主執行緒要結束了～");
}
```



9

- 執行緒欲繼承Thread以外的類別時，可以利用Runnable介面建立。
- Runnable介面只宣告一個run()方法，所以實作介面時只要實作run()方法即可。
- 實作Runnable介面的類別，還是必須依賴Thread類別的建構子才能建立一個執行緒物件。

10

- 使用Runnable型別物件建立Thread物件的建構子

使用Runnable物件的Thread建構子	說明
Thread(Runnable target)	以實作Runnable介面的類別物件target建立執行緒物件。
Thread(Runnable target, String name)	以實作Runnable介面的類別物件target建立執行緒物件，執行緒的名稱設定為name。

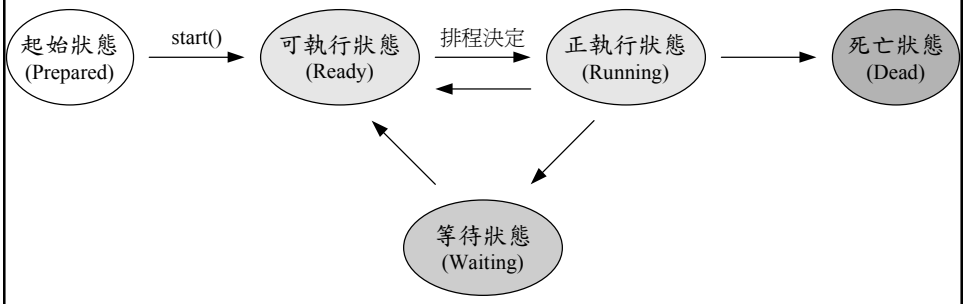
11

- 使用Runnable型別物件建立Thread物件，兩個物件是獨立的，不過可以看成是「Thread物件在啟動之後，呼叫Runnable型別物件的run()方法」。

```
final RunnableTest rt = new RunnableTest();
Thread mt = new Thread() {
        public void run() { rt.run(); } // 匿名類別
};
```

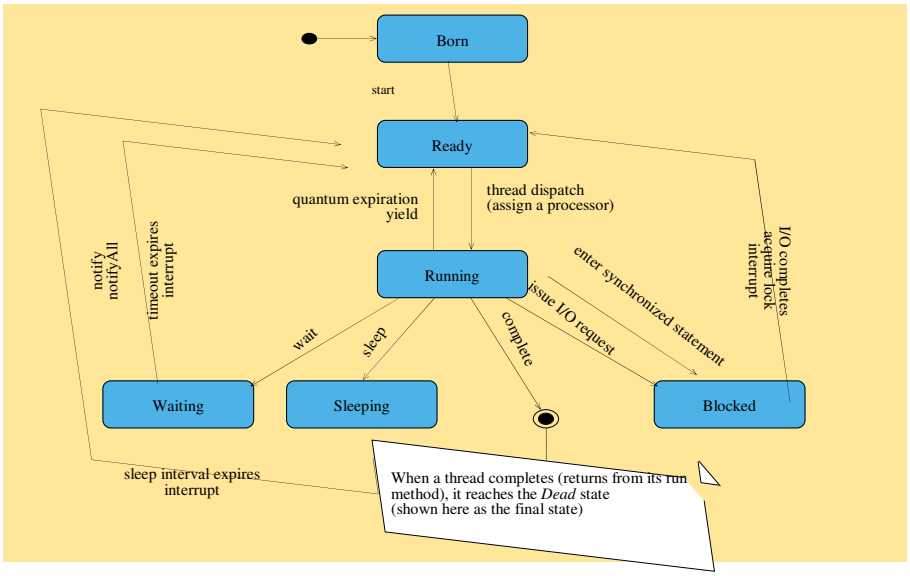
12

- 執行緒的狀態包括**起始狀態**(Prepared)、**可執行狀態**(Ready)、**正執行狀態**(Running)、**等待狀態**(Waiting)及**死亡狀態**(Dead)。



13

執行緒的生命狀態改變圖



- 執行緒各狀態的簡單說明：
  - **起始狀態**(Prepared)：當我們使用new關鍵字建立一個Thread物件後，未呼叫其start()方法。
  - **可執行狀態**(Ready)：正等著排程者(thread scheduler)安排執行。
  - **正執行狀態**(Running)：執行緒正在使用CPU的狀態
  - **等待狀態**(Waiting)：等待某事件的發生才繼續的狀態，例如呼叫sleep()方法。
  - **死亡狀態**(Dead)：執行緒執行完成後即進入死亡狀態。進入死亡狀態的執行緒不能被重新啟動。

15

12-5-1 放棄該次CPU的使用機會

- 為了避免CPU被獨佔，可以使用**yield()**方法讓某個執行緒進入Ready狀態，而暫時先讓出CPU。
- 呼叫高優先權執行緒的yield()方法，可以讓低優先權執行緒有較多的機會使用到CPU，以避免低優先權執行緒處於挨餓狀態(starvation)。

16

12-5-2 等待狀態

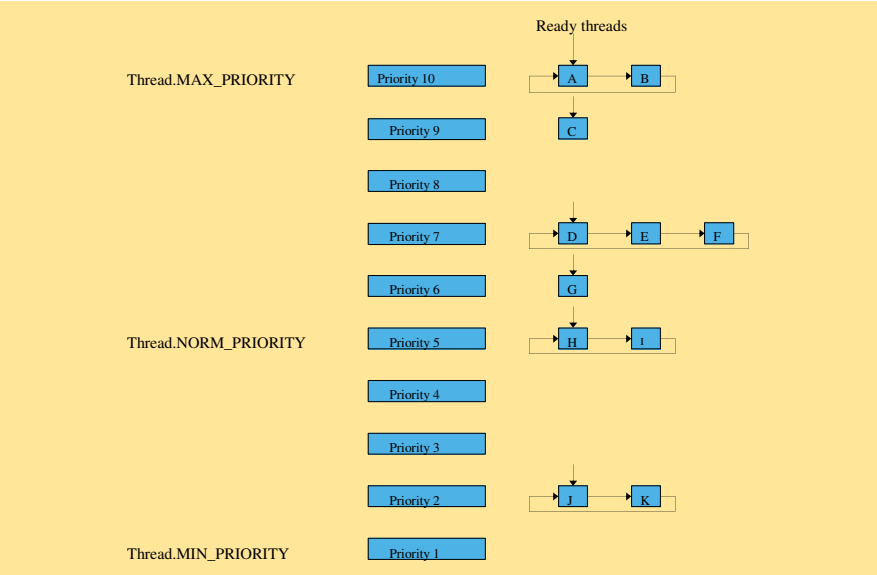
- 執行sleep()方法：暫停執行緒一段時間。
- 執行suspend()方法：沒有時間限制地暫停執行緒，可呼叫resume()方法回到Ready狀態。
- 呼叫join()方法：呼叫目標執行緒的join()方法，將等到目標執行緒結束之後才會繼續。
- Input/Output blocked：執行緒進行輸入輸出時，會因為輸入輸出的速度較慢而暫時進入Waiting。
- 呼叫同步方法時，未取得物件的lock。
- 執行wait()方法：放棄物件的使用權並進入等待狀態，可使用notify()方法喚醒執行緒。

17

- Java的多執行緒排程是使用簡單的「固定優先權排程」(fixed priority scheduling)，這種排程會依據可執行狀態之執行緒的優先權來決定順序。
- 子執行緒擁有和父執行緒相同的優先權。
- 執行緒建立之後，就可以使用getPriority()取得優先權值，以setPriority()方法來設定優先權值。
- Thread類別中定義了三個類別常數，
  NORM_PRIORITY是預設的優先權值，
  MIN_PRIORITY是最小的優先權值，
  MAX_PRIORITY為最大的優先權值。

18

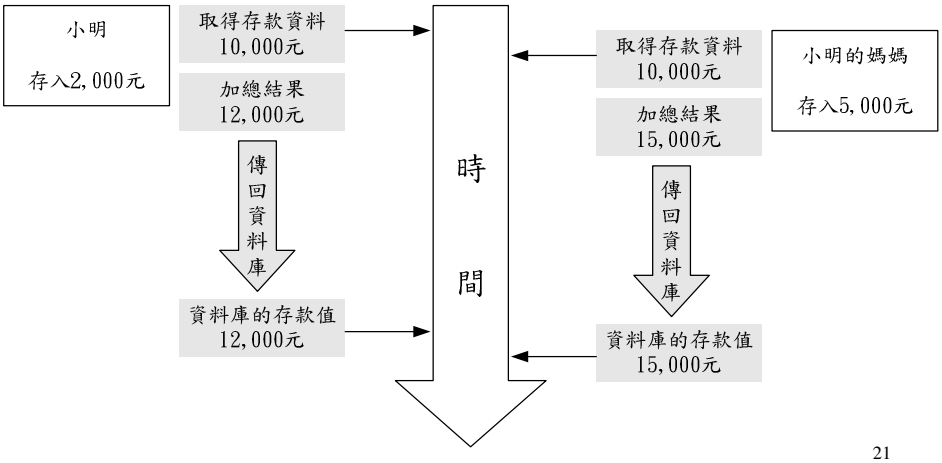
父執行緒相同的優先權排程範例



- 執行緒的排程實際上是和作業系統有關：
  - **先佔先贏**(preemptive scheduling)：高優先權的執行緒可持續執行，直至結束或等待，除非有更高優先權的執行緒出現。
  - **時間分配**(Time slicing)：CPU的使用被切成小段的時間，執行緒在使用過一單位的CPU時間後，就會進入Ready狀態，接著排程者會依優先權去挑選下一個執行者。優先權高的，有較大的機會執行。

12-7-1 使用共同資源的多執行緒

- 多執行緒可能造成的錯誤



21

12-7-2 同步化方法

- 使用synchronized關鍵字修飾**操作共同資源的方法**，可以讓該方法不會被兩個以上的執行緒同時使用，也就是在某個時間頂多只會有一個執行緒在使用該方法。
- 擁有synchronized修飾的方法之物件，就是一個**監視器**(Monitor)，監視器會讓物件內的synchronized方法只讓一個執行緒使用。
- 若物件中的所有方法都使用sysnchronized修飾，則在某個時間點只會有一個方法被執行。因為，**Monitor是物件而不是方法**。

22

12-7-3 synchronized敘述

- synchronized也可以當敘述使用，物件的同步

```
synchronized (物件) {
        //物件同步區
}
```

- synchronized區塊中的程式敘述還未執行完畢，該物件就不會被其它執行緒所使用。
- 當synchronized區塊在執行時，其同步化的物件的鎖(lock)是暫時被取用。

12-7-3 synchronized敘述

- 使用synchronized修飾方法和下式同義：

```
方法型別   方法名(形式參數列) {
        synchronized(this) {
                //方法內敘述
        }
}
```

- synchronized的同步化對象是物件實體，因此synchronized不能用來修飾**屬性**、**建構子**或**類別**。

12-7-4 wait()及notify()

- wait()、notify()和notifyAll()方法只能使用於synchronized修飾的方法或synchronized敘述中。
- wait()方法是讓執行緒進入等待的狀態(waiting pool)，同時「放棄物件的使用權」。
- wait()和sleep()兩者都可以讓執行緒進入等待狀態，不過sleep()並不會放棄物件的使用權。
- notify()會隨機叫醒waiting pool中的某個執行緒，而nitifyAll()則是叫醒waiting pool中所有的執行緒。

25

12-7-5 死結

- 在使用wait()及notify()時，必須注意避免讓執行緒進行到「**等待狀態**」之後出不來，如此造成執行緒永遠都在等，而且不會結束，這種情形稱為**死結**(deadlock)。
- 死結算是一種**邏輯上的錯誤**，發生死結時，並不會丟出例外，所以要格外小心避免。

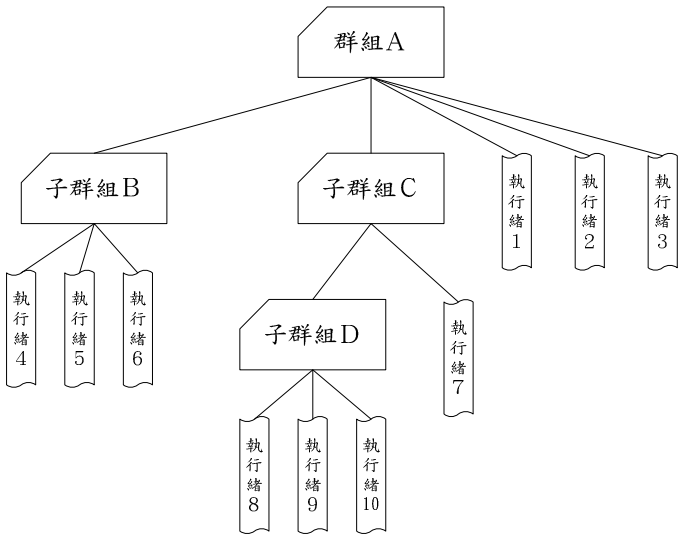
26

- ThreadGroup建構子

ThreadGroup的建構子	說明
ThreadGroup(String name)	建立一個名稱為name的執行緒群組物件。
ThreadGroup(ThreadGroup parent, String name)	在執行緒群組parent下，建立名為name的子執行緒群組。

27

- 執行緒群組可以包含子群組和執行緒



28

- 和執行緒群組相關的Thread建構子

執行緒加入群組的建構子	說明
Thread(ThreadGroup group, Runnable target)	以實作Runnable介面的target物件建立執行緒，並將執行緒歸於group群組。
Thread(ThreadGroup group, Runnable target, String name)	以實作Runnable介面的target物件建立名為name的執行緒，並將執行緒歸於group群組。
Thread(ThreadGroup group, String name)	建立名為name的執行緒，並將執行緒歸於group群組。

29

- 常用的ThreadGroup方法

ThreadGroup的方法	說明
final int **getMaxPriority**()	取得群組中最大的優先權值。
final String **getName**()	取得群組名稱。
final ThreadGroup **getParent**()	取得父群組。
final void **interrupt**()	中斷群組中所有的執行緒。
final void **setMaxPriority**(int pri)	設定群組的最高優先權值為pri。

30