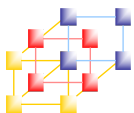


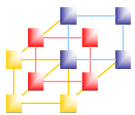
Unit 5

Concurrent Processes



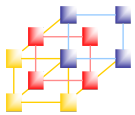
執行緒(thread)的觀念

- 電腦系統常利用多工(multitasking)的方式來提升效率
- 執行緒(thread)是一種輕量級(lightweight)的執行程式，占用較少的系統資源
 - 執行緒之間共享位址空間(address space)
 - 執行緒執行時的切換(context switching)成本較低
 - 執行緒之間的溝通方便



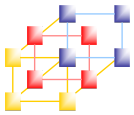
執行緒可分成兩大類

- 使用者執行緒(user threads)
 - 系統在執行應用程式時產生的執行緒 – main()
 - 稱為主執行緒(main thread)
- 常駐執行緒(daemon threads)
 - 系統產生的執行緒
 - 常駐於系統中，直到所有的使用者執行緒都結束才停止



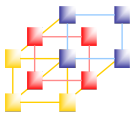
多執行緒程式設計的優點

- 資源的共用
- 提升系統回應的效率 – concurrent processing
- 整體效能的提升 – context switch 的負擔低



Python Thread

- Python提供了兩個模組
 - `_thread` – 低階模組
 - `threading` – 為高階模組，對`_thread`進行了封裝
- 啟動一個執行緒就是把一個函式傳入並建立 Thread 例項，然後呼叫`start()`開始執行
- `threading.Thread(target, name, args)`
 - `t1 = threading.Thread(target=func, args=(10,), name='Loop1')`
- `start()` – start the thread
 - `t1.start()`

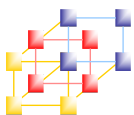


Thread class

```
class LoopThread(threading.Thread):
    def __init__(self, t_name, num):
        super().__init__(name=t_name)
        self.num = num

    def run(self):
        name = threading.current_thread().name
        print('Thread %s is running...' % name)
        for i in range(self.num):
            print(name, i)
            time.sleep(1)

if __name__ == '__main__':
    t1 = LoopThread('Loop1', 10)
    t1.start()
```

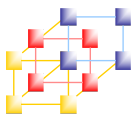


Thread methods

- `threading.active_count()`
 - Return the number of Thread objects currently alive
- `threading.current_thread()`
 - Return the current Thread object.
- `name`
 - A string used for identification purposes only
- `threading.main_thread()`
 - Return the main Thread object
- `join()`
 - Wait until the thread terminates
 - This blocks the calling thread until the thread whose `join()` method is called terminates

Network Programming

7



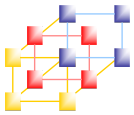
同時性控制 (Synchronization Control)

3-lock.py

- 當多個 Thread 會存取同一資源，而這一資源一次僅允許一個 Thread 存取時即需要同時性控制 (Critical Section Problem)
- `threading.Lock()`
 - `acquire()`
 - changes the state to locked and returns immediately
 - When the state is locked, `acquire()` blocks until a call to `release()` in another thread changes it to unlocked, then the `acquire()` call resets it to locked and returns
 - `release()`
 - should only be called in the locked state
 - it changes the state to unlocked and returns immediately
 - `locked()`
 - Return true if the lock is acquired

Network Programming

8

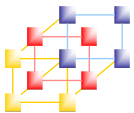


執行緒的溝通

- `threading.Condition()`
 - `wait(timeout=None)`
 - Wait until notified or until a timeout occurs.
 - `notify()`
 - By default, wake up one thread waiting on this condition, if any.
- If the calling thread has not acquired the lock when this method is called, a `RuntimeError` is raised
 - Use with statement


```
condition = threading.Condition()
with condition:
    condition.wait()
```

5-TCPClient.py
5-TCPServer.py



Multithread socket server

