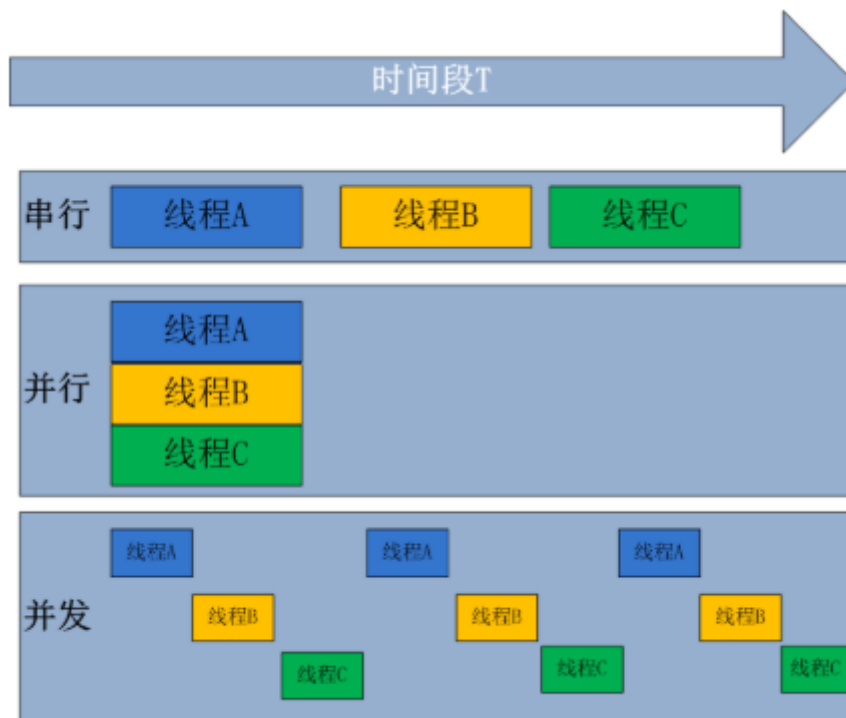


解释：一个进程包含一个或多个线程，线程共享进程的资源，cpu上跑的是线程



解释：并行就是同时处理多个任务，并发就是处理多个任务不一样要同时，并行是并发的子集

同步：请求【线程等待】---返回【线程继续执行】

异步：请求【线程继续执行】

```
pyqt5 D:\pycharm\pyqt5
build
dist
venv library root
apk_path
bg.jpg
cc.ico
cc.py
cc.spec
icon.jpg
name.txt
script_path
simulator_path
test.py
test.txt
TEST01.py
External Libraries
Scratches and Consoles

1  #!/usr/bin/python3
2
3  import _thread
4  import time
5
6
7  # 为线程定义一个函数
8  def print_time(threadName, delay):
9      count = 0
10     while count < 5:
11         time.sleep(delay)
12         count += 1
13         print("%s: %s" % (threadName, time.ctime(time.time())))
14
15
16     # 创建两个线程
17     try:
18         _thread.start_new_thread(print_time, ("Thread-1", 2,))
19         _thread.start_new_thread(print_time, ("Thread-2", 4,))
20     except:
21         print("Error: 无法启动线程")
22
23     while 1:
24         pass
25
26     print_time() > while count < 5

TEST01 x Turtle_PeppaPig x
D:\pycharm\pyqt5\venv\Scripts\python.exe D:/pycharm/pyqt5/TEST01.py
Thread-1: Tue Nov 5 19:17:11 2019
Thread-2: Tue Nov 5 19:17:13 2019
Thread-1: Tue Nov 5 19:17:13 2019
Thread-1: Tue Nov 5 19:17:15 2019
Thread-2: Tue Nov 5 19:17:17 2019
Thread-1: Tue Nov 5 19:17:17 2019
Thread-1: Tue Nov 5 19:17:19 2019
Thread-2: Tue Nov 5 19:17:21 2019
Thread-2: Tue Nov 5 19:17:25 2019
Thread-2: Tue Nov 5 19:17:29 2019
```

解释：尝试使用_thread去创建2个线程

```
pyqt5 / TEST01.py
Project ▾
  pyqt5 D:\pycharm\pyqt5
  build
  dist
  venv library root
  apk_path
  bg.jpg
  cc.ico
  cc.py
  cc.spec
  icon.jpg
  name.txt
  script_path
  simulator_path
  test.py
  test.txt
  TEST01.py
  External Libraries
  Scratches and Consoles

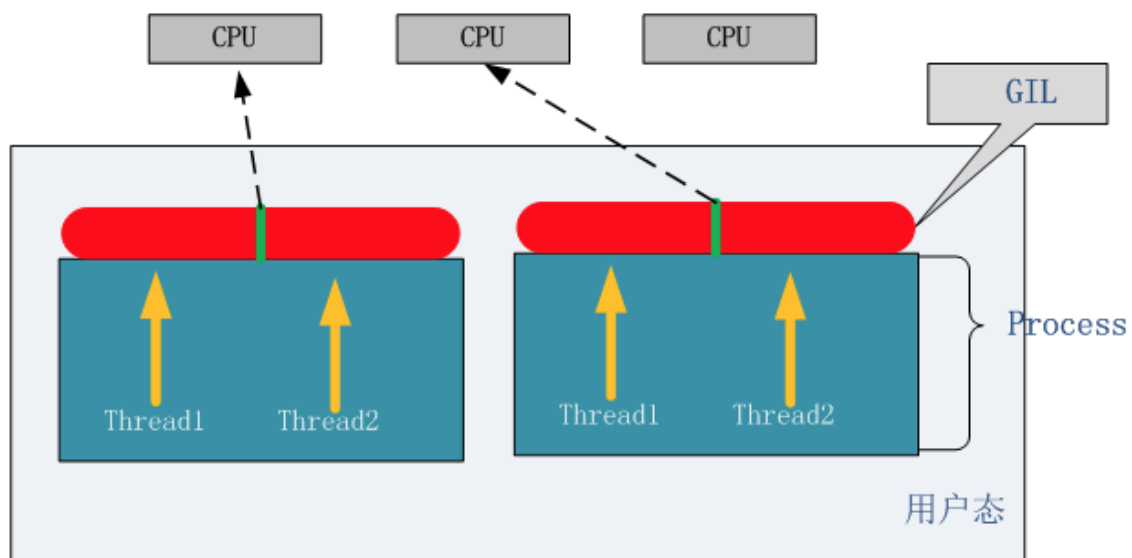
1  #!/usr/bin/python3
2
3  import threading
4  import time
5
6  exitFlag = 0
7
8  class myThread (threading.Thread):
9      def __init__(self, threadID, name, counter):
10         threading.Thread.__init__(self)
11         self.threadID = threadID
12         self.name = name
13         self.counter = counter
14     def run(self):
15         print("开始线程: " + self.name)
16         print_time(self.name, self.counter, 5)
17         print("退出线程: " + self.name)
18
19     def print_time(threadName, delay, counter):
20         while counter:
21             if exitFlag:
22                 threadName.exit()
23             time.sleep(delay)
24             print("%s: %s" % (threadName, time.ctime(time.time()
25                 counter -= 1
26
27     # 创建新线程
28     thread1 = myThread(1, "Thread-1", 1)
29     thread2 = myThread(2, "Thread-2", 2)
30
31     # 开启新线程
32     thread1.start()
33     thread2.start()
34     thread1.join()
35     thread2.join()
36     print("退出主线程")

Run: TEST01 ×
Thread-1: Wed Nov 6 10:01:11 2019
Thread-2: Wed Nov 6 10:01:12 2019
Thread-1: Wed Nov 6 10:01:12 2019
Thread-1: Wed Nov 6 10:01:13 2019
退出线程: Thread-1
Thread-2: Wed Nov 6 10:01:14 2019
```

解释：是用threading去管理线程，通过阻塞进程的方式去来实现多线程

```
cc.py × test.py × TEST01.py × Turtle_Peppapig.py ×
4 import time
5
6
7 class myThread(threading.Thread):
8     def __init__(self, threadID, name, counter):
9         threading.Thread.__init__(self)
10        self.threadID = threadID
11        self.name = name
12        self.counter = counter
13
14    def run(self):
15        print("开启线程: " + self.name)
16        # 获取锁，用于线程同步
17        threadLock.acquire()
18        print_time(self.name, self.counter, 3)
19        # 释放锁，开启下一个线程
20        threadLock.release()
21
22
23 def print_time(threadName, delay, counter):
24     while counter:
25         time.sleep(delay)
26         print("%s: %s" % (threadName, time.ctime(time.time())))
27         counter -= 1
28
29
30 threadLock = threading.Lock()
31 threads = []
32
33 # 创建新线程
34 thread1 = myThread(1, "Thread-1", 1)
35 thread2 = myThread(2, "Thread-2", 2)
36
37 # 开启新线程
38 thread1.start()
39 thread2.start()
```

解释：使用线程锁，解除线程锁，防止同一资源被多个线程访问和修改



结论：同一时刻同一进程中只能有一个线程被执行

解释：GIL全局解释器，cpython解释器引入的概念，同一时刻只能执行一个线程，保护全局解释器数据不被多个线程同时访问，用户自己的数据需要自己枷锁