day19-进程线程

- 概念

```
电脑
    - cpu 电脑中央处理器 相当于 人的大脑
    - 任务调度
        - 多个程序之间(进程)，采用 轮转式抢占时间片来执行任务的
        - 宏观并行，微观串行(单核一个cpu)
```

- 进程

```
进程：
    - 一个个正在运行的程序
    - pycharm typora sublime
    - 特点：
        - 独立性 （各个进程之间没有关联，互相独立）
        - 动态性
        - 结构性 数据资源 地址 空间
        - 并发性
```

- 线程

线程：

- 迅雷 多个下载任务
- 进程 包含 线程，有可能多个，有可能是单个。
- 线程是任务执行的基本单元，同时也是 资源分配基本单元
- 线程之间共享进程资源

- 线程 分类

1. 分类方式一
   - 如果一个进程中只有一个线程，可以称为 单线程
   - 如果一个进程中有多个线程，可以成为 多线程
2. 分类方式二
   - 主线程：启动程序，创建进程，会立即自动开启一个线程，称为主线程
   - 子线程：相对于主线程来说，其他线程 称为 子线程

- 线程 创建

1.方式一：

- 直接创建 需要继承自父类 threading.Thread，并且要重写run方法；实例对象通过调用start()方法 启动线程

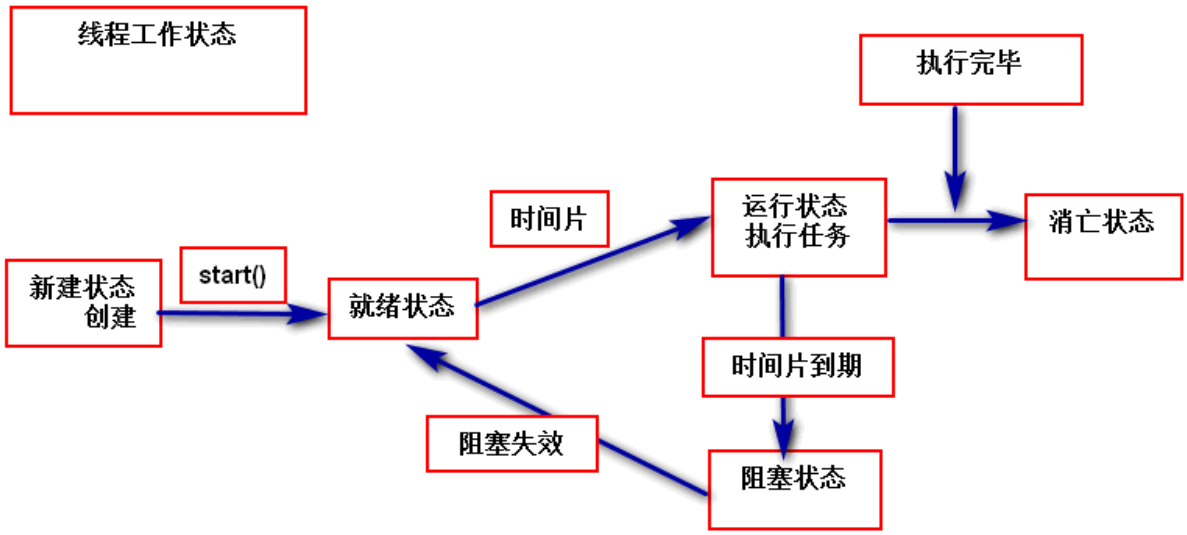
2.方式二：

- 使用父类 Thread 来创建线程，之后调用start()方法启动线程

```
def __init__(self, group=None, target=None,name=None,
                 args=(), kwargs=None, *, daemon=None)
```

- self： 类中进行方法创建时，默认有一个参数self
- group：预留参数
- target: 目标 表示当前线程所需要执行的目标任务
- name： 表示 线程名字
- args: 表示 目标任务所需要的参数，并且 元组形式
- kwargs: 表示 目标任务所需的参数 打包成字典形式
- daemon：设置是否为守护线程 bool值

- 线程状态图



- 线程对象常用方法属性

```
1. start()
    - 功能：启动线程
2. join([timeout])
    - 功能：调用该方法的线程，会阻塞当前主线程，只有在该线程任务执行完毕之后，才能再执行主线程。
    - timeout:可选参数 单位 秒
        - 功能：如果一旦达到timeout时间，则join失效。
3. setName(name)
    - 功能：设置线程的名字 尽量在 创建线程对象之后就调用该方法
```

- 注意：设置线程名字，也可通过 创建实例对象传参设置

4. setDaemon(bool)
   - 功能：设置线程为守护线程、精灵线程。
     当主线程运行完毕(当前进程内所有非守护线程都执行完毕之后，主线程才会消亡)消亡时，守护线程也会跟随消亡
     在 start()之前来设置
   - 参数：bool值
     - True：设置为守护线程
   - 注意：也可通过 创建线程对象时 传参来设置
   - 一般不重要的线程或者是需要依赖于主线程的 线程，会设置为守护线程，而一般操作人员不会对守护线程进行过多业务操作。
5. is_alive() / isAlive()
   - 判断某个线程是否存活
6. isDaemon()
   - 判断某个线程是否是守护线程
7. ident
   - 获取 当前线程的线程id
8. name
   - 获取 当前线程的 名字

- 线程同步

本质上是 保证数据的一致性，安全性
原子操作：不可分割的业务逻辑。
缘由：多个线程共享进程资源，在当前进程下操作同一数据(列表),有可能会出现数据相互修改，导致数据污染/数据脏读，从而让数据操作无效。而在数据操作过程中，实际是需要把 不可分割的业务代码 保护起来，防止多个线程同时修改产生的无效操作。
- 如何防止？需要 加锁。
- threading.Lock()
    - acquire() 获取/获得
    - release() 释放
    - 注意：一次原子操作中，只能申请一次，否则会出现死锁问题，相互等待现象。
- 为了解决死锁问题，可以使用RLock() 可重入锁
    - threading.RLock()

- GIL锁

```
GIL：
    - Global Interpreter Lock 全局解释器锁
    - 只针对Cpython解释器 （Jpython Ironpython pypy），Cpython解释器官方标准
    - 问题：规定了一个进程中只有抢到了GIL锁的线程 能执行任务。
                规定了一个进程中只能有一个线程执行任务。
    - 多个线程 争抢的是 GIL锁 （等同于 时间片）
    - 弊端：
            - 导致 python下 多线程 变成了 单线程
            - 降低了 多核 的使用 性能，不能发挥 多核优势作用
    - 优势：
        - 线程切换要快速，并且节省资源开销。
        - I/O密集型操作可以应用。文件读写，网络请求处理
```

- 进程&多进程

进程：一个一个正在运行的程序
特点：结构性 独立性 动态性 并发性
分类：
    - 单进程
    - 多进程：
        - 多个进程之间相互独立
        - 不受GIL影响，不需要考虑单核限制问题。
        - 如果多核情况下，可以实现真正的并行
        - 如果有计算要求高的任务，可以使用多进程。计算 视频编

- 进程创建

可以类比 线程创建。
1. 使用模块 multiprocessing
    multiprocessing.Process()
    - Process() 使用方式，与 线程类使用方式相同，并且也有同名的方法 属性 供使用
    - start() 开启进程
    - join()
    - 设置 守护进程 没有对应方法设置，可以通过 创建对象 传参设置
    - 设置name 可以通过 创建对象 参数设置

- 进程池

pool: 池子
进程池：创建进程时，可以批量创建。 （普通方式可以，只不过python有简洁方式，使用pool）

当有新的任务要执行时，如果进程池里进程已满，则需要等待，等待当前进程池中某个进程执行任务完毕，再加入到进程池中进行任务执行。

1. Pool()
2. apply_async()
3. apply()
5. terminate() 中断

异步：

-当前某一个进程1正在执行某个数据操作任务，此时其他进程也想要执行某个任务，可以进行，不需要等到进程1执行完毕之后。

同步：

- 当前某一个进程1正在执行某个数据操作任务，此时其他进程也想要执行某个任务，不能进行，必须等到进程1执行完毕之后，才能执行。

- 多进程下常用通信方法

```
1.Queue

2.pipe
```

- 生产者消费者模型

```
```