Python_并发 多进程或多线程

特别注意：之所以要使用concurrent.futures的原因是：该模块在多进程和多线程之间的切换特别容易

```
import concurrent.futures,collections
import urllib.request
import os

URLS = ['http://www.foxnews.com/',
        'http://www.cnn.com/',
        'http://europe.wsj.com/',
        'http://www.bbc.co.uk/',
        'http://some-made-up-domain.com/']

Result = collections.namedtuple("Result","url content")#具名元组，多用于存储结构化数据

# Retrieve a single page and report the url and contents
def load_url(url, timeout=2):
    conn = urllib.request.urlopen(url, timeout=timeout)
    return Result(url,conn.readall())


if __name__=="__main__":

    with concurrent.futures.ProcessPoolExecutor(max_workers=os.cpu_count()) as executor:
#将ProcessPoolExecutor替换为ThreadPoolExecutor即可实现多线程
        fs = [executor.submit(load_url,url) for url in URLS]#存储任务队列
        for future in concurrent.futures.as_completed(fs):#每次迭代取一个已经完成的任务
            err = future.exception()#如果某个人物已经停止，且其执行过程中抛出异常
            if not err:
                result = future.result()#依次取并行执行任务结果
                print('Handled url:{}'.format(result.content))
            else:#打印异常信息
                print('generated an exception')
```