

并发执行

本章中描述的模块支持并发执行代码。适当的工具选择取决于要执行的任务（CPU密集型或IO密集型）和偏好的开发风格（事件驱动的协作式多任务或抢占式多任务处理）。这是一个概述：

- [threading --- 基于线程的并行](#)
 - 概述
 - GIL 和性能的考量
 - 参考
 - 线程局部数据
 - 线程对象
 - Lock 对象
 - RLock 对象
 - Condition 对象
 - Semaphore 对象
 - Semaphore 示例
 - Event 对象
 - Timer 对象
 - Barrier 对象
 - 在 with 语句中使用锁、条件和信号量
- [multiprocessing --- 基于进程的并行](#)
 - 概述
 - Process 类
 - 上下文和启动方法
 - 在进程之间交换对象
 - 进程间同步
 - 进程间共享状态
 - 使用工作进程
 - 参考
 - Process 和异常
 - 管道和队列
 - 杂项
 - 连接对象 (Connection)
 - 同步原语
 - 共享 ctypes 对象
 - `multiprocessing.sharedctypes` 模块
 - 管理器
 - 自定义管理器
 - 使用远程管理器
 - 代理对象
 - 清理

- 进程池
- 监听器及客户端
 - 地址格式
 - 认证密码
 - 日志记录
 - `multiprocessing.dummy` 模块
- 编程指导
 - 所有start方法
 - `spawn` 和 `forkserver` 启动方式
- 例子
- `multiprocessing.shared_memory` --- 可跨进程直接访问的共享内存
- `concurrent` 包
- `concurrent.futures` --- 启动并行任务
 - Executor 对象
 - ThreadPoolExecutor
 - ThreadPoolExecutor 例子
 - InterpreterPoolExecutor
 - ProcessPoolExecutor
 - ProcessPoolExecutor 例子
 - Future 对象
 - 模块函数
 - Exception 类
- `concurrent.interpreters` --- 同一进程中的多个解释器
 - 关键细节
 - 概述
 - 多解释器与隔离
 - 在一个解释器中运行
 - 并发与并行
 - 解释器间通信
 - "共享"对象
 - 参考
 - 解释器对象
 - 异常
 - 解释器间通信
 - 基本使用
- `subprocess` --- 子进程管理
 - 使用 `subprocess` 模块
 - 常用参数
 - `Popen` 构造函数
 - 异常
 - 安全考量
 - `Popen` 对象
 - Windows `Popen` 助手

- Windows 常数
- 较旧的高阶 API
- 使用 `subprocess` 模块替换旧函数
 - 替代 `/bin/sh` shell 命令替换
 - 替代 shell 管道
 - 替代 `os.system()`
 - 替代 `os.spawn` 函数族
 - 替代 `os.popen()`
- 旧式的 Shell 发起函数
- 备注
 - 超时行为
 - 在 Windows 上将参数列表转换为一个字符串
 - 禁用``posix_spawn()``
- `sched` --- 事件调度器
 - 调度器对象
- `queue` --- 同步队列类
 - Queue对象
 - 等待任务完成
 - 终结队列
 - SimpleQueue 对象
- `contextvars` --- 上下文变量
 - 上下文变量
 - 手动上下文管理
 - `asyncio` 支持

以下是上述某些服务的支持模块：

- `_thread` --- 低层级多线程 API