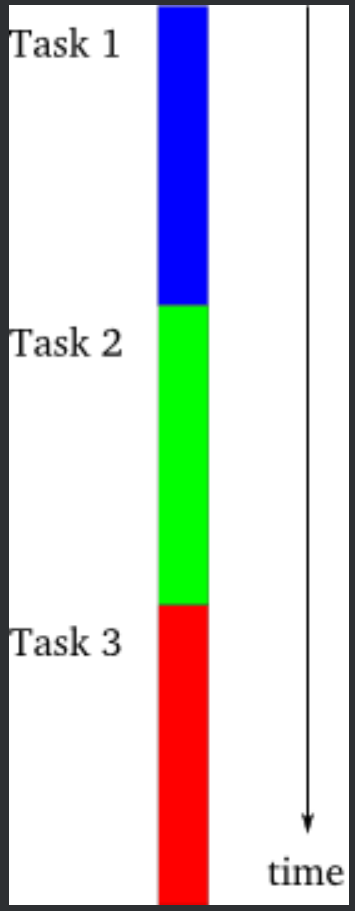
# 异步编程和事件驱动

# 同步/异步

同步和异步描述的是进程/线程的调用方式

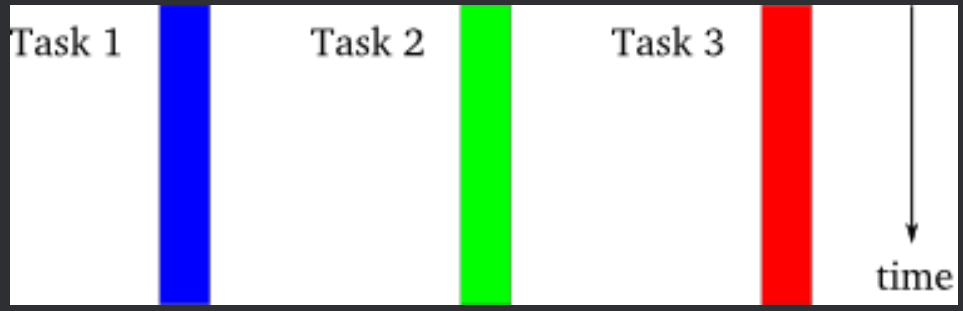
1. 同步调用指的是线程发起调用后，一直等待调用返回后才继续执行下一步操作，这并不代表 CPU 在这段时间内也会一直等待，操作系统多半会切换到另一个线程上去，等到调用返回后再切换回原来的线程
2. 异步就相反，发起调用后，线程继续向下执行，当调用返回后，通过某种手段来通知调用者



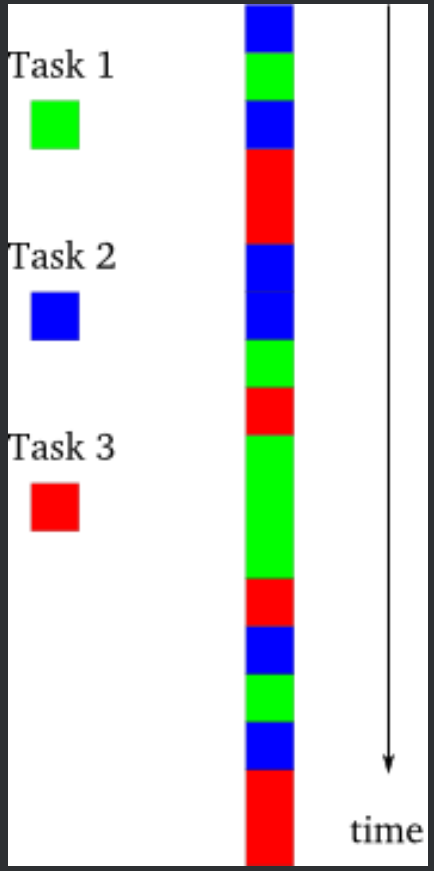

# 单线程的同步模型

在一个时刻，只能有一个任务在执行，并且前一个任务结束后一个任务才能开始。如果任务都能按照事先规定好的顺序执行，最后一个任务的完成意味着所有任务都完成

# 多线程/多进程的同步模型

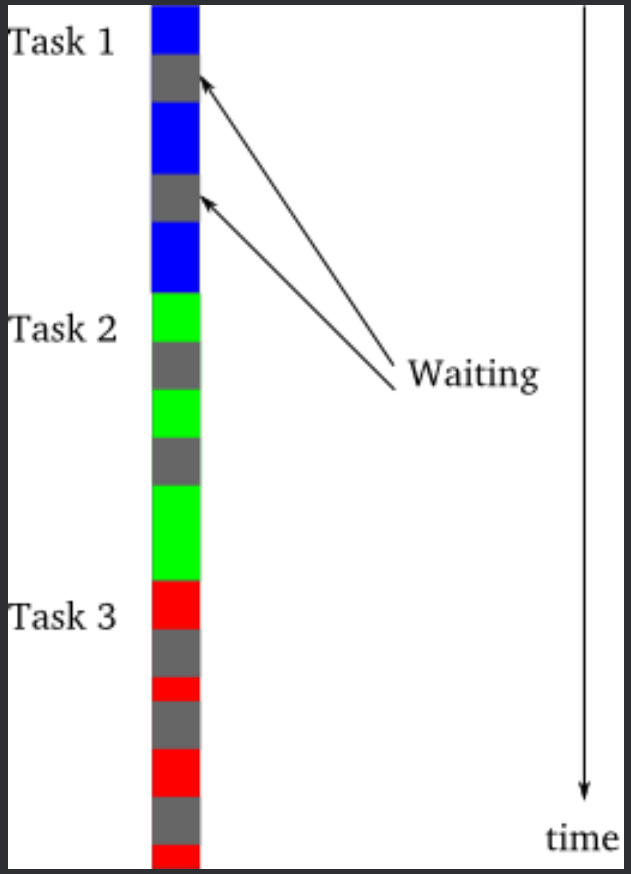



👆每个任务都在单独的线程(进程)中完成

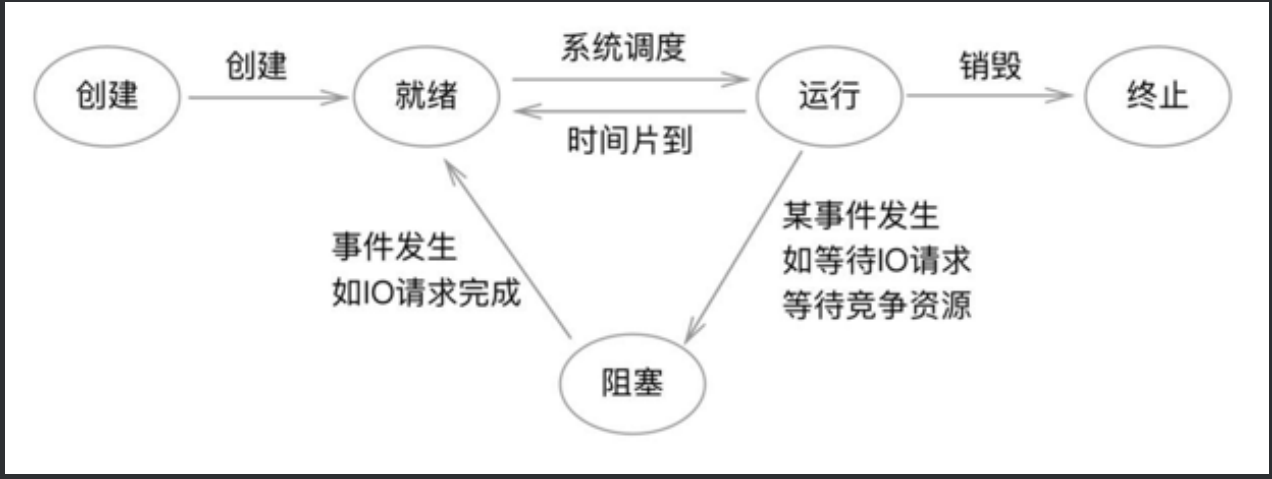



# 异步编程模型

1. 单线程
2. 任务交错执行


Task 1
Task 2
Task 3
Waiting
time

# 阻塞/非阻塞




阻塞与非阻塞的概念是针对 IO 状态而言的，关注程序在等待 IO 调用返回这段时间的状态

# IO 编程模型

1. blocking I/O  阻塞IO
2. non-blocking I/O  非阻塞IO
3. I/O multiplexing I/O复用
4. signal driven I/O 信号驱动式IO
5. asynchronous I/O 异步IO

# 如下场景能发挥异步模型的优势👇

1. 有大量的任务，因此在一个时刻至少有一个任务要运行
2. 任务执行大量的I/O操作，这样同步模型就会在因为任务阻塞而浪费大量的时间
3. 任务之间相互独立，以至于任务内部的交互很少

# 事件驱动模型

事件驱动模型主要应用在图形用户界面、网络服务和Web前端上。举个编写图形用户界面程序的例子，要给界面上每一个按钮都添加监听函数，而该函数则只有在相应的按钮被用户点击的事件发生时才会执行，开发者并不需要事先确定事件何时发生，只需要编写事件的响应函数即可。监听函数或者响应函数就是所谓的事件处理器(event handler)，类似的事件还有鼠标移动、按下、松开、双击等等，这就是事件驱动。

事件驱动的程序一般都有一个主循环（main loop）或称事件循环(event loop)，该循环不停地做两件事：事件监测和事件处理。首先要监测是否发生了事件，如果有事件发生则调用相应的事件处理程序，处理完毕再继续监测新事件。事件循环只是在一个进程中运行的单个线程

# Python 2 标志性的异步框架

1. Tornado
2. Twisted
3. Gevent

# 延伸阅读

1. http://krondo.com/in-which-we-begin-at-the-beginning/
2. 《UNIX网络编程卷1：套接字联网API》 第6章
3. 《深入理解计算机系统》 第11章