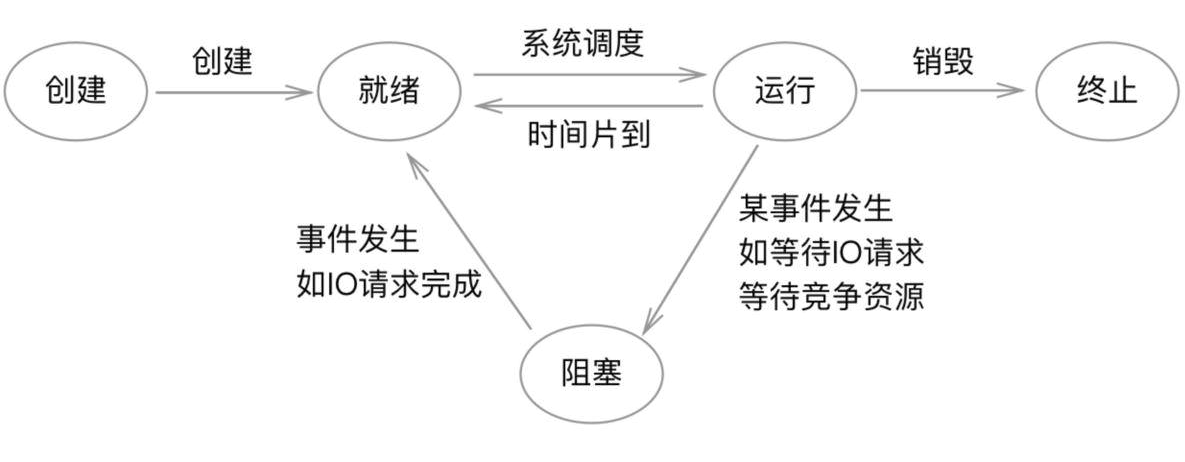
**多线程(重)**

## 阻塞/非阻塞

* 阻塞调用：调用结果返回之前，调用者会进入阻塞状态等待，只有在得到结果后才会返回
* 非阻塞调用：在不能立刻得到结果之前，该函数不会阻塞当前线程，而是立刻返回



## 同步/异步

* 同步：在发出一个同步调用时，在没有得到结果之前，该调用就不返回
* 异步：在发出一个异步调用后，调用者不会立即得到结果，该调用就返回了

**同步例子**

int n = func();

next();

// func() 的结果没有返回，next() 就不会执行，直到 func() 运行完。

**异步例子**

func(callback);

next();

...

void callback(int n) // func 结果回调

{

int k = n;

}

// func() 执行后，还没得出结果就立即返回，然后执行 next() 了

// 等到结果出来，func() 回调 callback() 通知调用者结果。

* 同步和阻塞区别：同步调用，线程不一定阻塞，线程可能还在运行状态。但阻塞的话线程肯定是不在运行状态了。
* 异步和非阻塞区别：异步是说调用的时候结果不会马上返回，线程可能阻塞了也可能没阻塞，两者无必然联系。非阻塞是指调用时线程肯定不会进入阻塞状态。

上面两组概念，就有4种组合。  
同步阻塞调用：得不到结果不返回，线程进入阻塞态等待。  
同步非阻塞调用：得不到结果不返回，线程不阻塞一直在CPU运行。  
异步阻塞调用：去到别的线程，让别的线程阻塞起来等待结果，自己不阻塞。  
异步非阻塞调用：去到别的线程，别的线程一直在运行，直到得出结果。

## 并发/并行

* 并发：一段时间内，有几个程序（可能是几个进程的线程，也可能是一个进程的多个线程）在同个CPU（即单核）上运行，但任意时刻点上只有一个程序在处理机上运行。
* 并行：一段时间内，有几个程序（可能是几个进程的线程，也可能是一个进程的多个线程）在多个CPU（即多核）上运行，任意时刻有多个程序运行，且它们互不干扰。

