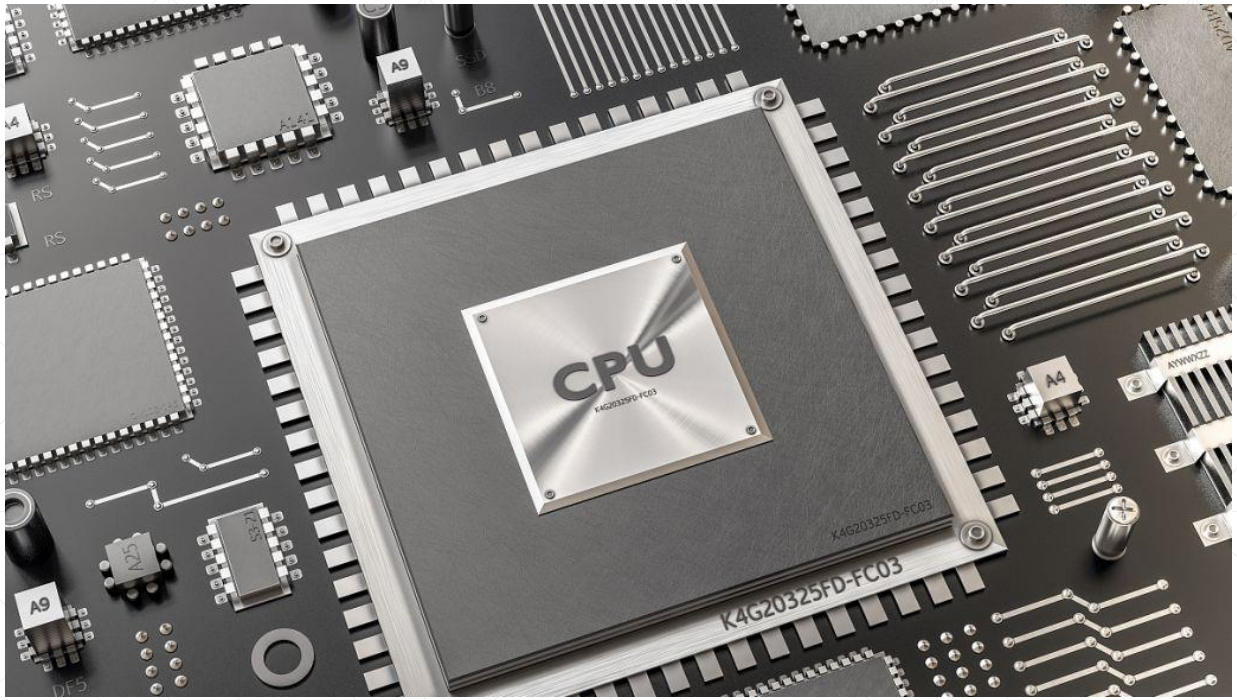


进程与线程

北京理工大学计算机学院
金旭亮

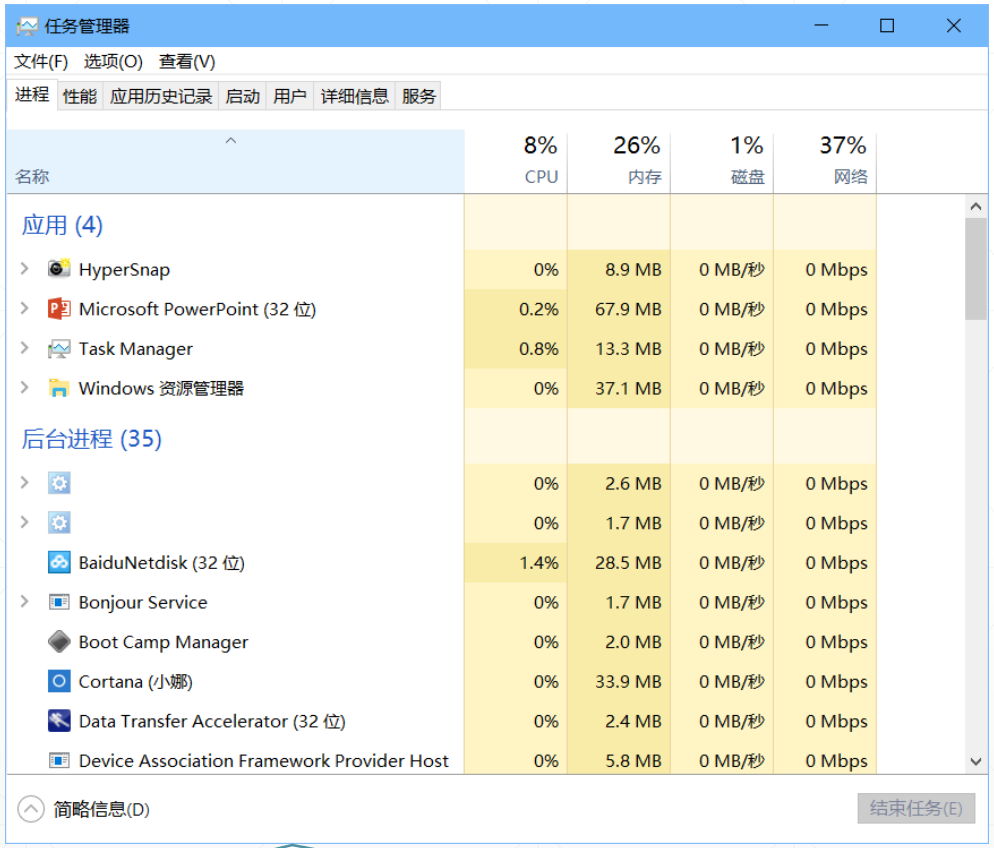
导引：让我们从“线程”开始……

- 在其他的课程中介绍过Java多线程开发的基础知识。有这些知识作为基础，掌握Android多线程技术几乎没有什么太大的难度，很多知识都是共通的。

- 与Java SE应用类似，Android也使用Thread对象代表一个线程，同样可以采用Java SE标准的编程方式编写多线程代码。

- Kotlin对线程进行了一些扩展，从而使代码变得更为精简。另外，Kotlin还提供了一种“轻量级”的线程，称为“**协程（Coroutine）**”，我们将在后面的课程中介绍它。当前，只介绍线程。

理解“进程”的概念



任务管理器可以列出当前正在运行的所有进程。

每个正在运行的程序，计算机都将其视为一个“**进程（Process）**”。

进程是操作系统分配各种资源（比如内存）的基本单位。

进程在运行过程中，可以创建多个“**线程（Thread）**”，第一个创建的线程，通常是“**主线程（Main Thread）**”。

进程的销毁

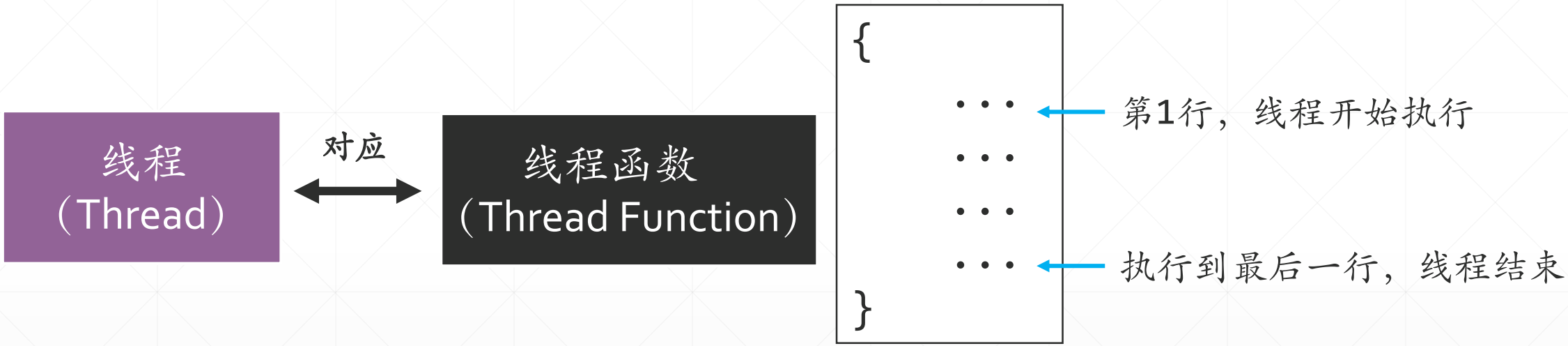
- 一个正在运行的Android App，对应着一个进程，它在运行过程中所创建的所有对象实例（比如Activity，Fragment，各种控件），都归属于这个进程。

- 当资源紧张时，Android操作系统会销毁某些后台进程，当一个进程被销毁时，这个进程所创建的所有组件，也一并被销毁。

什么叫线程?

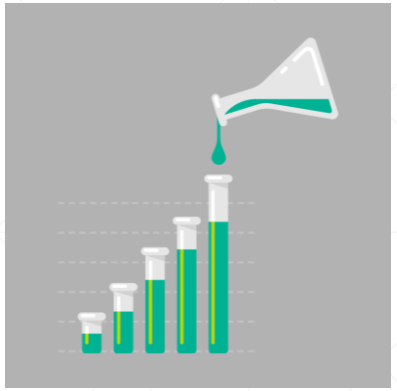
线程就是“正在执行中的一段代码”，这段代码，完成特定的数据处理工作。

线程与线程函数



每个线程都对应着一个线程函数，线程的执行，也就是线程函数的执行。

线程的创建与启动



Kotlin标准库提供了简便的方法用于启动一个线程:

```
Thread {
    //需要以多线程方式运行的代码
}.start()
```

进一步简化

```
thread {
    //需要以多线程方式运行的代码
}
```

thread函数创建的线程会自动运行，并且是一个前台线程。

使用Kotlin标准库函数创建并启动线程

main.kt

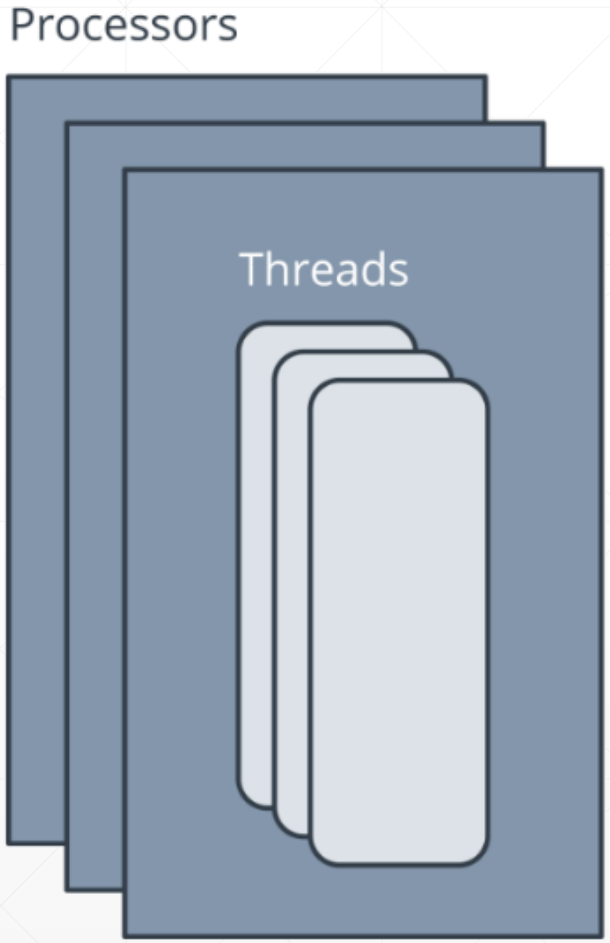
```kotlin
import kotlin.concurrent.thread

fun main(args: Array<String>) {
    //自定义新线程名字，并自启动
    thread(start = true,name="自定义工作线程"){
        (1..3).forEach { it: Int
            println("线程 ${Thread.currentThread().name} 正在处理 $it")
        }
    }
    (1..3).forEach { it: Int
        println("线程 ${Thread.currentThread().name} 正在处理 $it")
    }
}
```

Run: MainKt

```
"C:\Program Files\Java\jdk-15\bin\java.exe"
线程 main 正在处理 1
线程 自定义工作线程 正在处理 1
线程 main 正在处理 2
线程 自定义工作线程 正在处理 2
线程 main 正在处理 3
线程 自定义工作线程 正在处理 3

Process finished with exit code 0
```

IntelliJ Kotlin示例项目：WhatIsThread

CPU与多线程

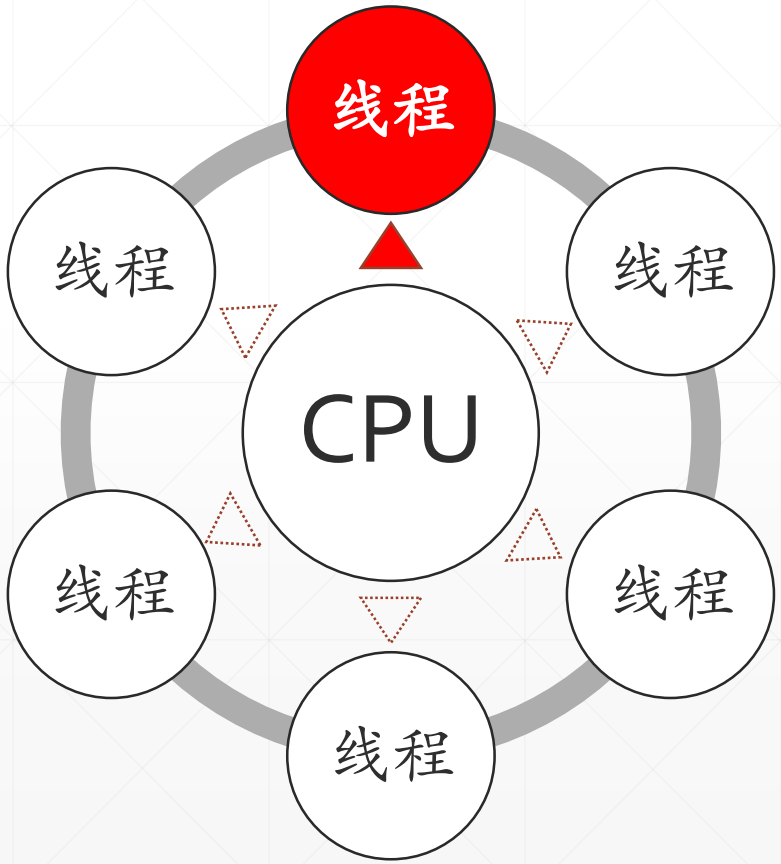


一台计算机，至少拥有一个CPU，当前的主流CPU，基本上都是多核的，每个CPU的执行核，在特定的时间片内，都可以运行一个线程。

多核CPU，实现了数据的并行处理。

操作系统使用“时间片”来给线程分配CPU

拥有“CPU”使用权的线程居于运行状态。

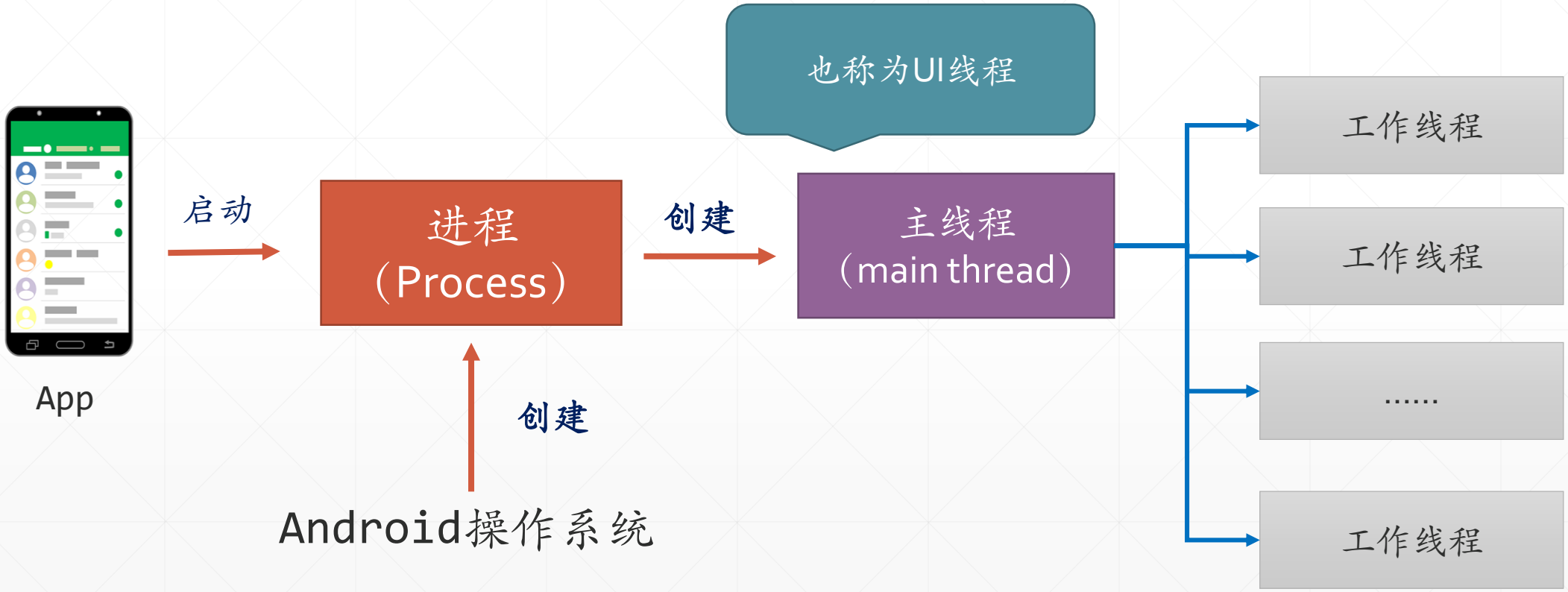


宏观上“并行”，微观上“串行”

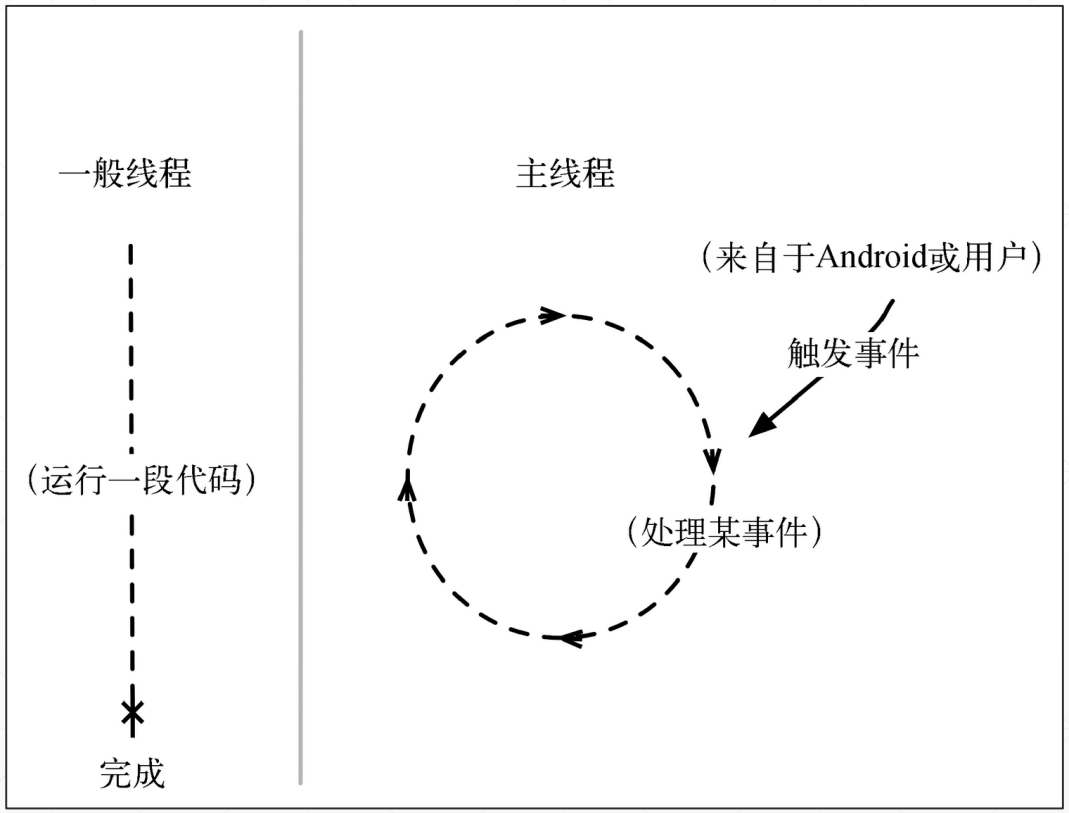
其实每个CPU的执行核，一次只能运行一个线程，但由于这些线程占用CPU的时间很短，所以，在一个“比较长”的时间段内，从外部来看，所有的线程都在运行中，是“并行”运行的。

在哪个时刻选择哪个线程运行，由操作系统所实现的线程调度算法决定，程序员可以通过一些手段（比如设定线程优先级），对这一过程施加影响。

Android平台上的进程与线程



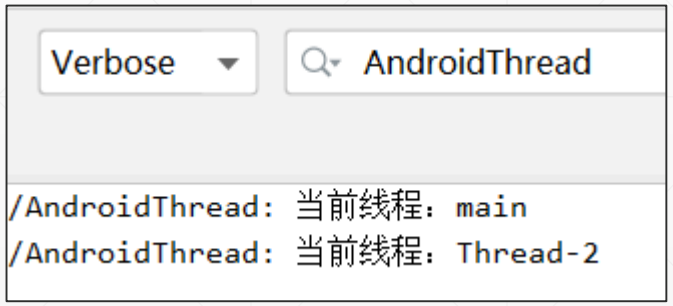
两种类型的Android线程



示例：AndroidThread

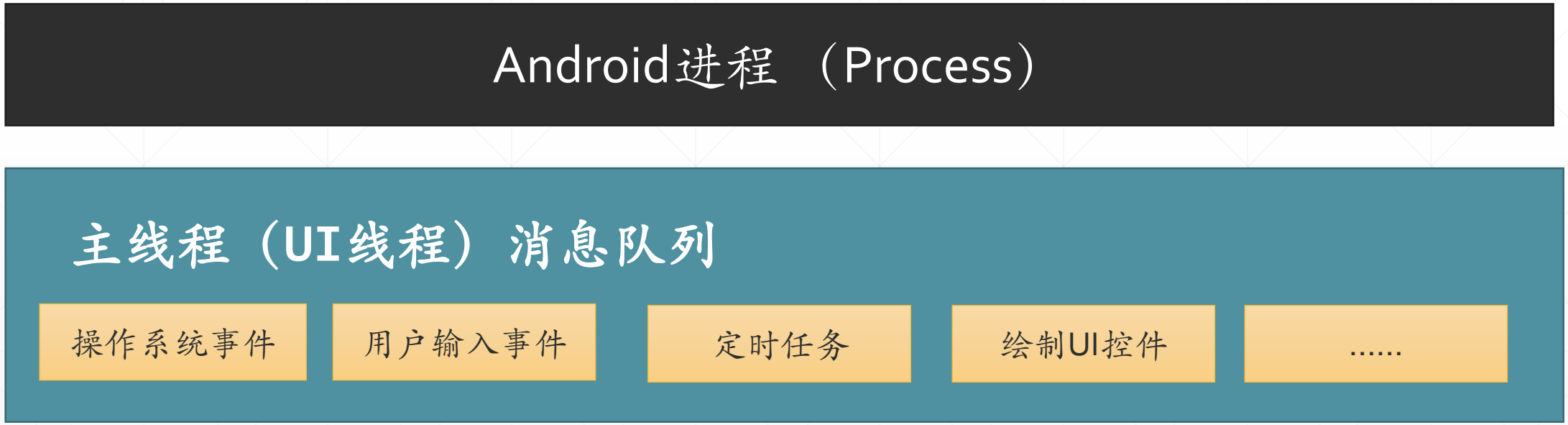
```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        log("当前线程：${Thread.currentThread().name}")
        Thread {
            log("当前线程：${Thread.currentThread().name}")
        }.start()
    }
}
```

运行截图



Android App中的主线程

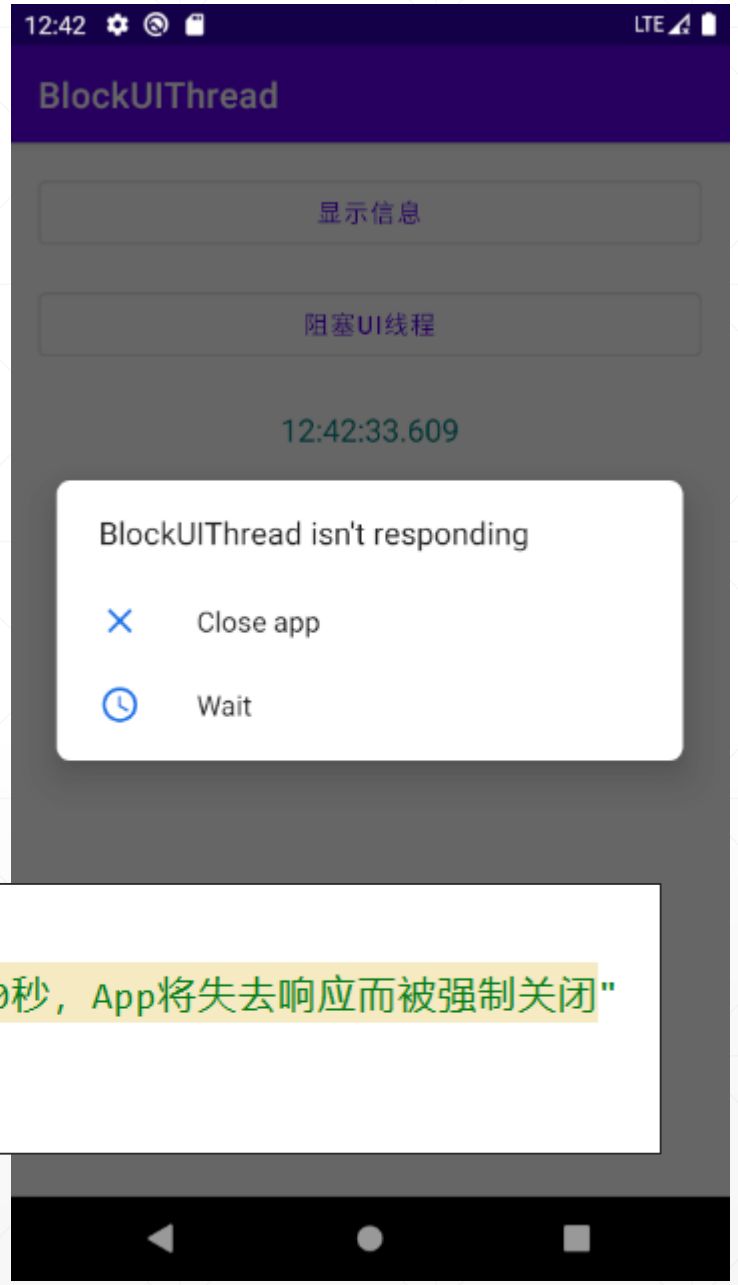
每个Android 应用都有一个**主线程**，负责绘制界面、协调用户互动以及接收生命周期事件等工作，职责非常之多……



Android控件重绘原理

如果你让主线程干了太多的活，那么App就无法及时地响应用户的操作，因为主线程这时正忙于干你给它指定的活.....


```
fun sleepThreeSeconds(){
    tvInfo.text = "主线程休眠3秒，按钮将无法点击……"
    Thread.sleep( millis: 3000)
}
```




```
fun generateANRError(){
    tvInfo.text = "主线程休眠30秒，App将失去响应而被强制关闭"
    Thread.sleep( millis: 30000)
}
```

小结：Android App开发基本原则

- 每个Android 应用的**主线程**，它不应该完成太多的工作，尤其不能被阻塞。
- 任何长时间运行的计算和操作（例如解码位图、访问磁盘或执行网络请求），都应该创建工作线程，在后台完成。