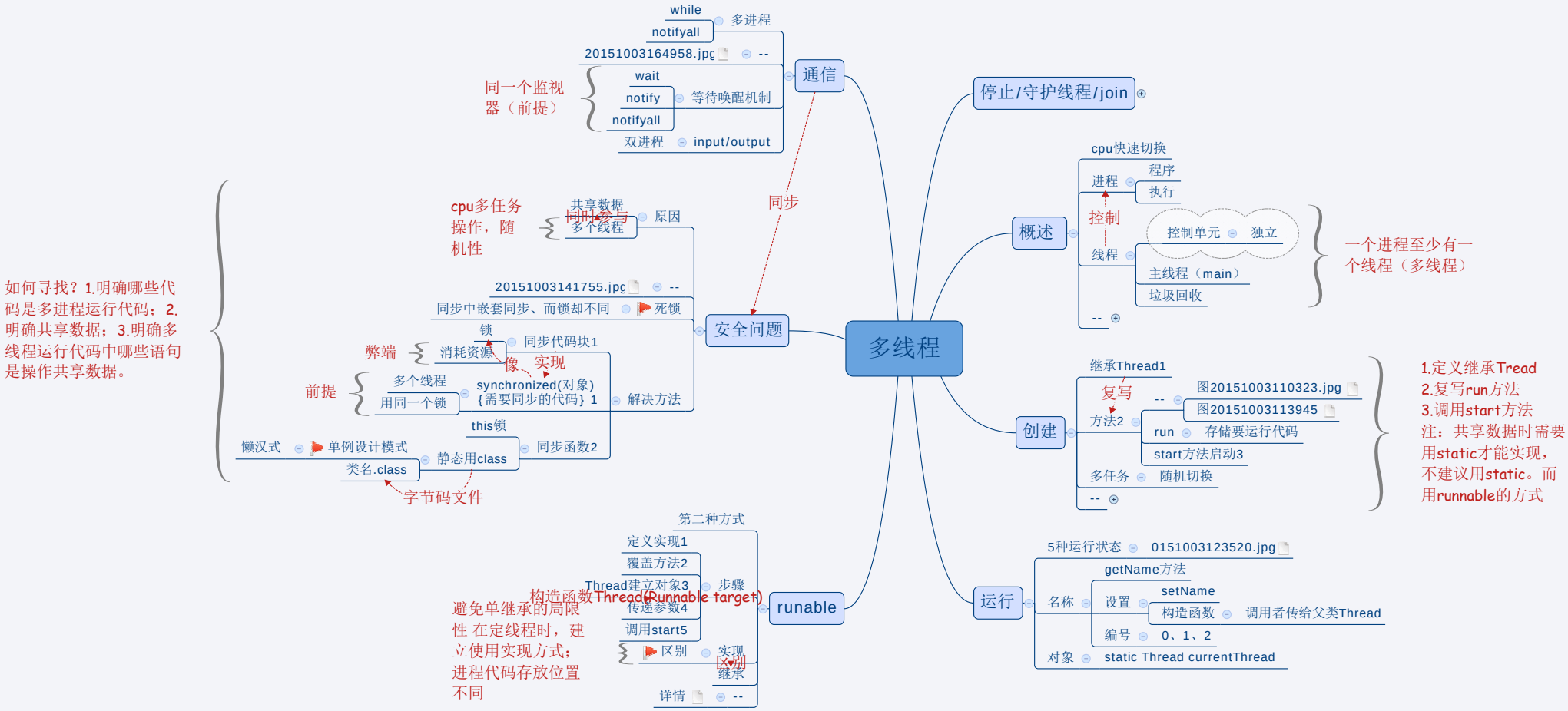
多线程
通信
多进程
while
notifyall
20151003164958.jpg
--
等待唤醒机制
wait
notify
notifyall
同一个监视器（前提）
双进程
input/output
同步
安全问题
原因
共享数据
同时参与
多个线程
cpu多任务操作，随机性
20151003141755.jpg
--
死锁
同步中嵌套同步、而锁却不同
解决方法
同步代码块1
锁
消耗资源
弊端
像
实现
synchronized(对象)
{需要同步的代码} 1
多个线程
用同一个锁
前提
同步函数2
this锁
静态用class
单例设计模式
懒汉式
类名.class
字节码文件
如何寻找？1.明确哪些代码是多进程运行代码；2.明确共享数据；3.明确多线程运行代码中哪些语句是操作共享数据。
runable
步骤
第二种方式
定义实现1
覆盖方法2
Thread建立对象3
传递参数4
调用start5
构造函数Thread(Runnable target)
区别
实现
继承
区别
详情
--
避免单继承的局限性 在定线程时，建立使用实现方式；进程代码存放位置不同
停止/守护线程/join
概述
cpu快速切换
进程
程序
执行
控制
线程
控制单元
独立
主线程（main）
垃圾回收
--
一个进程至少有一个线程（多线程）
创建
继承Thread1
复写
方法2
--
图20151003110323.jpg
图20151003113945
run
存储要运行代码
start方法启动3
多任务
随机切换
--
1.定义继承Tread
2.复写run方法
3.调用start方法
注：共享数据时需要用static才能实现，不建议用static。而用runnable的方式
运行
5种运行状态
0151003123520.jpg
名称
getName方法
设置
setName
构造函数
调用者传给父类Thread
编号
0、1、2
对象
static Thread currentThread