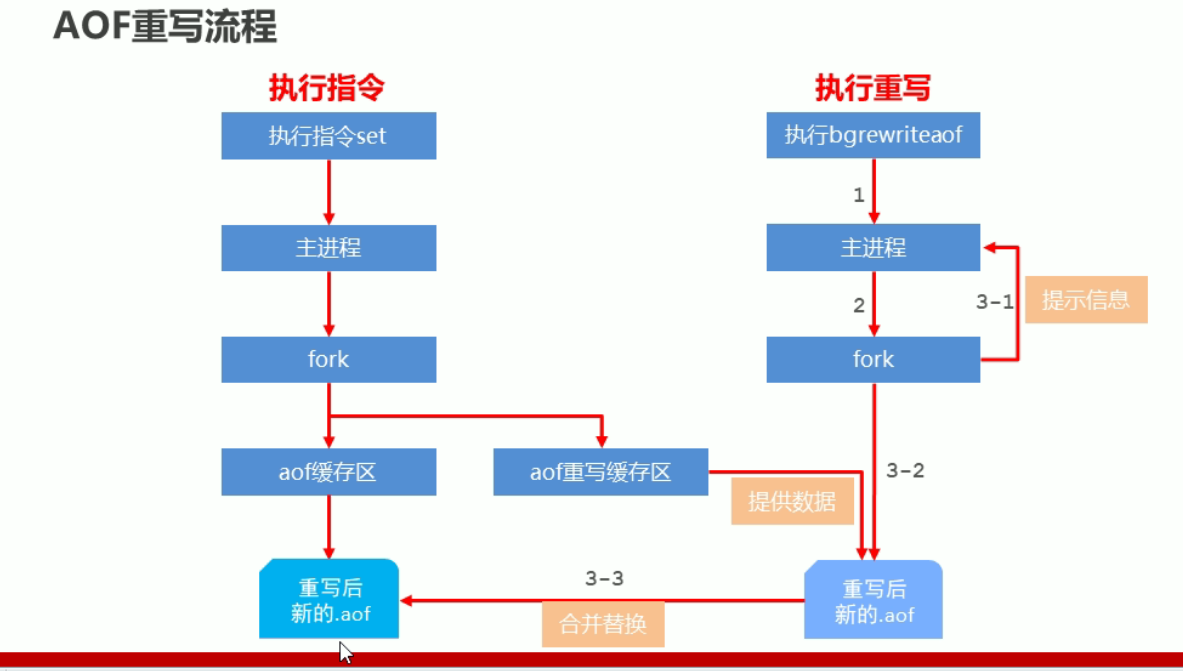
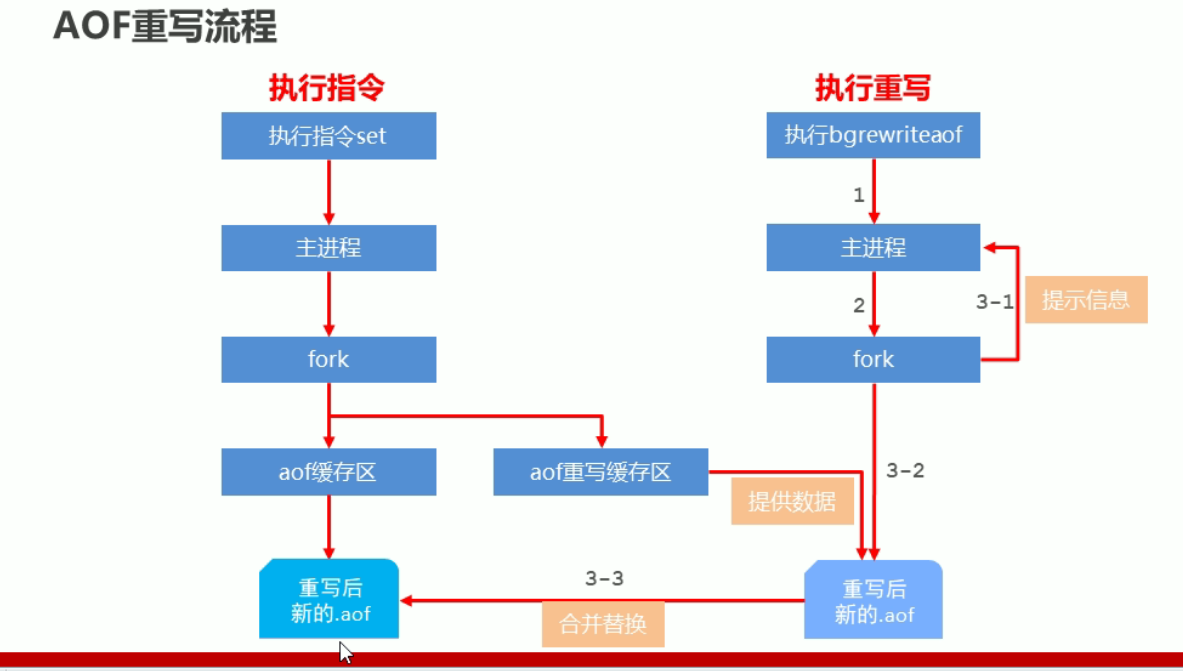
# redis

1. AOF与RDB的区别
   1. 无法承受数分钟以内的丢失，可用AOF
   2. 能承受，且要求数据集的恢复速度，选用RDB
   3. 区别
      1. AOF存储速度比RDB快
      2. AOF恢复速度比RDB要慢，因为AOF存的是过程呀，还需要执行
      3. 安全性就得看AOF选用哪种策略了，RDB是一定会丢失数据的
      4. AOF资源消耗比RDB低
      5. AOF优先级大于RDB
2. AOF持久化
   1. 记录数据的过程（当数据需要恢复时，将过程重新执行一遍即可）
   2. 解决数据持久化的实时性（主流）
   3. AOF写数据的过程
      1. 客户端发送指令，它不会立即执行，会先向指令存入缓冲区，生成一个AOF文件，到达一定阶段就会将指令同步到AOF文件中
      2. 至于什么时候同步提供了三个策略
         1. alaways（每次）
            1. 每次都会同步进AOF文件，数据0误差，但是效率低呀
         2. everysec（每秒）
            1. 每秒会同步一次，数据准确性高，丢也就丢一秒嘛，性能较高，建议（默认）
         3. no（系统控制）
            1. 由操作系统控制的，整体不可控
      3. appendonly yes|no 开启持久化配置（默认关闭）
      4. appendfsync AOF的写数据策略
   4. AOF写数据遇到的问题
      1. 如果同一个key设置多次，那么前几次是不是属于无效的数据呢，就没必要执行
      2. 重写
         1. 作用：提高持久化效率，数据恢复效率，磁盘利用率
         2. 规则
            1. 进程内超时的数据就不写入AOF文件了
            2. 忽略无效指令，只保留最终数据的写入命令
            3. 对同一数据的多条写入命令，合并为一条
   5. AOF工作原理
      1. eveysec开启重写的流程
         1. 首先执行set指令，进入主进程开启一个子进程fork
         2. 在存进缓存区的时候，会开出一个aof重写的缓存区，它为重写提供数据
         3. 执行bgrewriteaof指令，进入主进程，开启一个子进程，它会发一段提示信息，表示正在执行
         4. 然后会创建一个重写后的aof文件，里面的数据是由之前的缓存区提供的
         5. 最后和执行set指令那段子进程的aof文件进行替换





* 1. AOF的启动有持久化写策略和手动重写和自动配置重写

1. RDB持久化
   1. 存储量大，效率低
   2. 基于快照思想，bgsave会开一个子进程
   3. 丢失数据，因为快照思想是记录某个时间点的数据
2. 事务
   1. 事务的执行流程
      1. 写一个指令set，它会判断你在不在事务状态，怎么判断呢，它会识别你这个指令，因为你是set普通指令，所以直接执行，返回结果
      2. 如果说是multi开启事务，那么就不是一个普通指令了，它会创建一个队列queue，返回一个ok，接下来你再输入一个set普通指令的话，它会把你这个放入队列中，返回一个queued已在队列的信息，直到你输入的是exec执行事务，才会将队列一起执行，然后返回结果
      3. 如果执行了discard，取消事务，那么就会销毁这个队列，返回一个ok
   2. 事务的注意事项
      1. 没有那个自动回滚，只能靠自己在代码中实现手动回滚，最好是不要敲错
      2. 基于特定条件的事务执行-锁
         1. watch 对key'添加一个监视锁，在执行exec之前发生变换，就会终止这个事务执行
3. 过期键的删除策略
   1. 惰性过期
      1. 只有当访问一个key时，才会判断key是否已经过期，可能会出现大量过期key没有呗访问，也不会被清理
   2. 定期过期
      1. 每隔一段时间会定期去清理这个过期的key