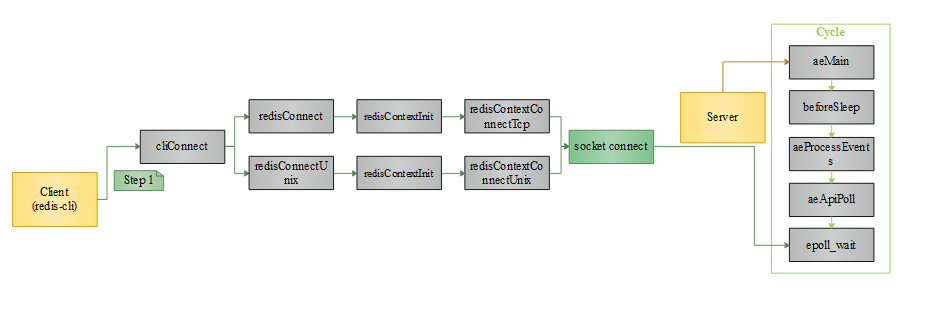
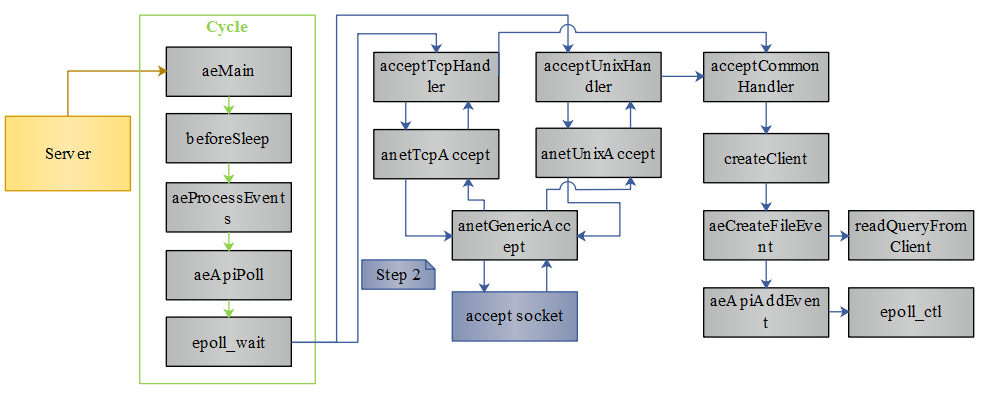
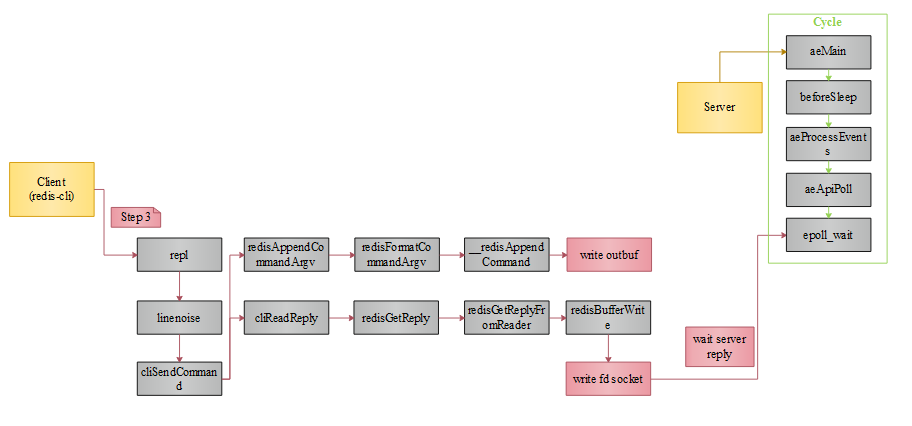
Client 发起socket 连接：



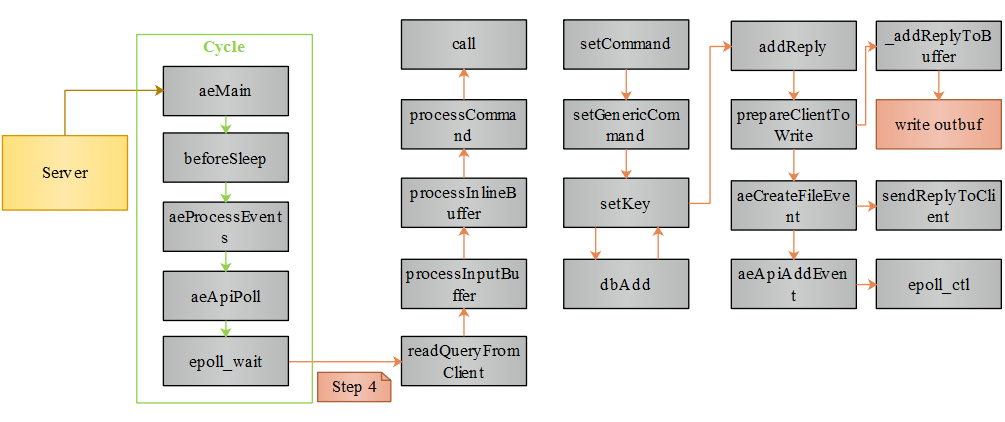
Server 接受socket连接：



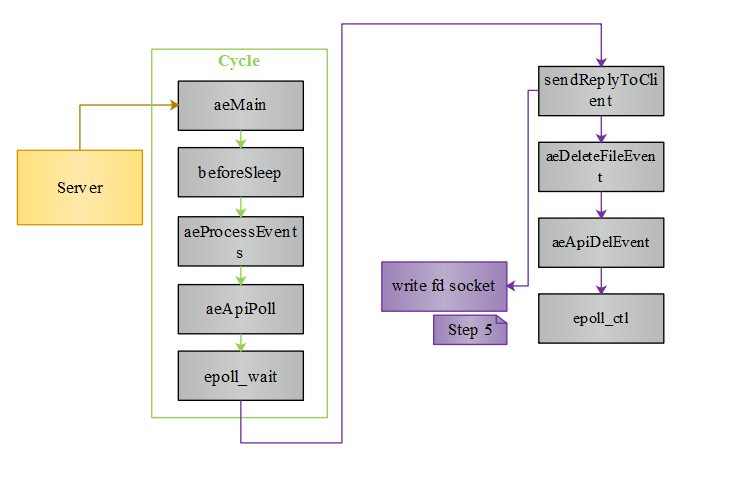
客户端开始写入：



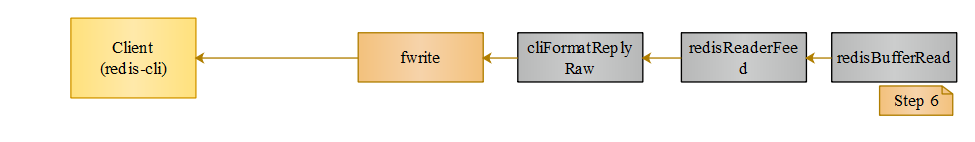
server 端接收写入：



server 返回写入结果：



Client收到返回结果：



1. 建立连接

redis在将监听socket初始化完毕之后，会将他们添加到事件循环中(server.c)：

**for** **(**j **=** 0**;** j **<** server**.**ipfd\_count**;** j**++)** **{**

**if** **(**aeCreateFileEvent**(**server**.**el**,** server**.**ipfd**[**j**],** AE\_READABLE**,**

acceptTcpHandler**,NULL)** **==** AE\_ERR**)**

**{**

redisPanic**(**

"Unrecoverable error creating server.ipfd file event."**);**

**}**

**}**

其中，networking.c/acceptTcpHandler函数处理客户端建立连接，并返回客户端对应的socket。然后，调用networking.c/acceptCommonHandler函数进行初始化。

acceptCommonHandler函数主要调用networking.c/creatClient函数初始化客户端相关data struct及对应的socket。

creatClient函数首先创建redisClient，设置socket属性。然后添加该socket的读事件。将socket设置为nonBlock和tcpNoDelay，最后注册该socket的读事件。事件处理函数是networking.c/readQueryFromClient，这个函数便是客户端请求的起点。

1. 请求处理
2. readQueryFromClient函数读取请求buffer

redis请求buffer分为inline和multibulk两种类型。

readQueryFromClient函数首先，设置当前服务的client，然后是设置这次从socket读取的数据的默认大小（REDIS\_IOBUF\_LEN为16KB）。然后，重新设置读取数据的大小，避免频繁拷贝数据。通过调用sds.c/sdsMakeRoomFor函数，将读取的请求内容储存到redisClient->querybuf中。接下来调用read系统调用，读取readlen大小的数据，并存储到querybuf中。最后，调用networking.c/ processInputBuffer函数解析请求。

1. processInputBuffer函数解析请求

根据clien->reqtype判断请求类型，分别调用processInlineBuffer函数和processMultibulkBuffer函数解析请求，其结果储存于client->argv数组和client->argc变量中。

processInlineBuffer函数和processMultibulkBuffer函数返回REDIS\_OK，调用server.c/processCommand函数命令处理。

1. 服务器执行命令

processCommand函数主要逻辑：首先通过c->argv[0]在命令字典中找到对应命令(调用sds.c/lookupCommand函数)，然后做一系列判定(auth验证等)。跳过后面很大一部分校验代码，最后，调用server.c/call函数，回调该命令的处理函数。

Call函数中，真正调用命令执行函数的代码为：

c->cmd->proc(c)

c->cmd是一个redisCommand结构变量，其中的proc函数可在server.c中的redisCommandTable中找到

1. 发送响应

在请求处理完之后，进行响应时，需要调用addReplyXXX族函数。

redis将响应内容组织成两部分，一个固定大小的buffer（16KB），一个响应内容对象的链表。在链表为空并且buffer有足够空间时，则将响应添加到buffer中，否则创建一个节点追加到链表上。固定buffer和响应链表，整体上构成了一个队列。这么组织的好处是，既可以节省内存（不需一开始预先分配大块内存，动态内存的由链表cover），并且可以避免频繁分配、回收内存（在响应内容小于buffer大小时）。

AddReply函数首先调用prepareClientToWrite函数初始化，prepareClientToWrite函数在每次发送响应时调用，并且要在向响应buffer添加数据之前。只有在返回REDIS\_OK时，才能向响应buffer输出内容。在响应buffer为空时（固定buffer以及链表均为空）时，向事件循环注册写事件，回调函数是sendReplyToClient。

接下来，需要将响应内容添加到output buffer中。总体思路是，先尝试向固定buffer添加，添加失败的话，在尝试尾插法添加到响应链表。如果尾节点和当前响应内容大小之和小于REDIS\_REPLY\_CHUNK\_BYTES（16KB），则合并成一个节点。

最后，sendReplyToClient函数发送buffer。