**一、介绍**  
  
            Redis客户端使用RESP（Redis的序列化协议）协议与Redis的服务器端进行通信。 虽然该协议是专门为Redis设计的，但是该协议也可以用于其他 客户端-服务器 （Client-Server）软件项目。  
  
            RESP是对以下几件事情的折中实现：  
  
                1、实现简单  
  
                2、解析快速  
  
                3、人类可读  
  
            RESP可以序列化不同的数据类型，如整数（integers），字符串（strings），数组（arrays）。它还使用了一个特殊的类型来表示错误（errors）。请求以字符串数组的形式来表示要执行命令的参数从客户端发送到Redis服务器。Redis使用命令特有（command-specific）数据类型作为回复。  
  
            RESP协议是二进制安全的，并且不需要处理从一个进程传输到另一个进程的块数据的大小，因为它使用前缀长度（prefixed-length）的方式来传输块数据的。  
         （RESP is binary-safe and does not require processing of bulk data transferred from one process to another, because it uses prefixed-length to transfer bulk data.）  
  
            注意：该文章所说的协议是仅用于 **客户端 - 服务器**（Client-Server）的通信。 Redis集群使用不同的二进制协议来交换节点之间的消息。  
  
  
**二、Redis协议的详解**  
  
          要想更好的使用Redis，如果没有对Redis的协议更深的了解，要想精通恐怕很难，现在我们就来看看Redis的协议是什么。  
  
**1、网络层（Networking layer）**  
  
                         客户端连接到Redis的服务器，创建到端口6379的TCP连接。  
  
                         尽管，RESP协议是非TCP专用的技术，但在Redis的环境中，该协议仅用于TCP连接（或类似于Unix套接字的面向流的连接）。  
                         While RESP is technically non-TCP specific, in the context of Redis the protocol is only used with TCP connections (or equivalent stream oriented connections like Unix sockets).  
  
  
           **2、请求-响应模型（Request-Response model）**  
  
                       Redis接受由不同参数组成的命令。 一旦接收到命令，它就会被处理并且发送响应回客户端。  
  
                       这是最简单的模式，但也有两个例外的情况：  
  
                             1、Redis支持管道操作（稍后会在本文档中介绍）。所以客户可以一次发送多个命令，稍后等待回复。  
  
                             2、当Redis客户端订阅 Pub/Sub模式的通道时，协议会改变语义变成推送协议，也就是说，客户端不再需要发送命令，因为服务器一旦收到消息就会自动向客户端发送该新消息（对于订阅了通道的客户端）。  
  
                        除了上述两个例外，Redis协议就是一个简单的 请求-响应 协议。  
  
  
**3、RESP协议描述（RESP protocol description）**  
  
                        RESP协议在Redis 1.2版本中引入，但它已成为在Redis 2.0版本中与Redis服务器沟通的标准方式。这是您应该在Redis客户端中实现的协议。  
  
                        RESP实际上是一个支持以下数据类型的序列化协议：简单字符串（Simple Strings），错误（Errors），整数（Integers），块字符串（Bulk Strings）和数组（Arrays）。  
  
                        在Redis中,RESP用作 请求-响应 协议的方式如下：  
  
                            1、客户端将命令作为批量字符串的RESP数组发送到Redis服务器。  
  
                            2、服务器（Server）根据命令执行的情况返回一个具体的RESP类型作为回复。  
  
                        在RESP协议中，有些的数据类型取决于第一个字节：  
  
                            1、对于简单字符串，回复的第一个字节是“+”  
  
                            2、对于错误，回复的第一个字节是“ - ”  
  
                            3、对于整数，回复的第一个字节是“：”  
  
                            4、对于批量字符串，回复的第一个字节是“$”  
  
                            5、对于数组，回复的第一个字节是“\*”  
  
                    此外，稍后会讲RESP协议能够使用指定的 Bulk Strings 或Array 的特殊变量来表示空值。  
  
                    在RESP协议中，协议的不同部分始终以“\r\n”（CRLF）结尾。  
  
  
**4、RESP简单字符串（RESP Simple Strings）**  
  
                   简单字符串按以下方式编码：以+（加号字符）开始，后跟一个不能包含CR或LF字符的字符串（不允许换行符），以CRLF（即“\r\n”）结尾。  
  
                   简单字符串用于以最小开销传输非二进制安全的字符串。例如，许多Redis命令在成功时回复“OK”，因为RESP Simple String使用以下5个字节进行编码：

"+OK\r\n"

                  为了发送二进制安全的字符串，需要使用RESP Bulk Strings。  
  
                  当Redis以简单字符串回复时，客户端库应该返回给调用者一个由'+'后的第一个字符组成的字符串，直到字符串结尾，不包括最终的CRLF字节。  
  
  
**5、RESP错误（RESP Errors）**  
  
                  RESP协议针对错误具有特定数据类型表示。实际上，错误与RESP Simple Strings完全相同，但第一个字符是减号' - '而不是加号。简单字符串和RESP错误之间的真正区别在于错误被客户端视为异常，而组成错误类型的字符串本身就是错误信息。  
  
                  基本格式是：

"-Error message\r\n"

                  错误回复仅在发生错误时发送，例如，如果您尝试针对错误的数据类型执行操作，或者命令不存在等等。 当收到错误应答时，客户端就应该抛出一个异常。  
  
                  以下是错误回复的示例：

-ERR unknown command 'foobar'

-WRONGTYPE Operation against a key holding the wrong kind of value

                  “ - ”之后的第一个单词，直到第一个空格或换行符，表示返回的错误种类。这只是Redis使用的一种约定，并不是RESP错误格式的一部分。  
  
                  例如，ERR是通用错误，而WRONGTYPE是一个更具体的错误，意味着客户端试图针对错误的数据类型执行操作。 这被称为错误前缀，并且是一种允许客户端了解服务器返回的错误类型而不依赖于给定的确切消息的方式，该消息可能随时间而改变。  
  
                  客户端实现可能会针对不同的错误返回不同类型的异常，或者可能会提供一种通用方法来通过直接将错误名称作为字符串提供给调用者来捕获错误。  
  
                  然而，这样的功能不应该被认为是至关重要的，因为它很少有用，而针对客户端有限的实现来说可能仅仅返回一个通用错误条件，例如 false 。  
             
  
**6、RESP整数（RESP Integers）**  
  
                 这种以“：”字节为前缀，并且只是以一个CRLF终止字符串的类型就表示是整数。 例如“：0\r\n”或“：1000\r\n”是整数回复。  
  
                 许多Redis命令返回RESP整数，如 INCR，LLEN 和 LASTSAVE。  
  
                 返回的整数没有特殊含义，它只是INCR的增量数，LASTSAVE的UNIX时间等等。但是，返回的整数保证位于有符号的64位整数范围内。  
  
                 整数回复也广泛用于返回true或false。例如像 EXISTS 或 SISMEMBER 这样的命令将返回1为真，0为假。  
  
                 其他命令如 SADD，SREM 和 SETNX 将在实际执行操作时返回1，否则返回0。  
  
                 以下命令将回复一个整数回复：SETNX，DEL，EXISTS，INCR，INCRBY，DECR，DECRBY，DBSIZE，LASTSAVE，REINENX，MOVE，LLEN，SADD，SREM，SISMEMBER，SCARD。  
  
  
**7、RESP大容量字符串（RESP Bulk Strings）**  
  
                  大容量字符串用于表示长达512 MB的单个二进制安全字符串。  
  
                  **大容量字符串按以下方式编码：**  
  
                       1、一个以“$”字节开始，后面是组成字符串的字节数（前缀长度），由CRLF终止。  
  
                        2、实际的字符串数据。  
  
                       3、最终的CRLF。  
  
                    所以字符串“foobar”被编码如下：

"$6\r\nfoobar\r\n"

                   当一个空字符串只是：

"$0\r\n\r\n"

                 还可以使用RESP Bulk Strings 的特殊格式来表示空值。在这种特殊的格式中，长度是-1，并且没有数据，所以空值表示为：

"$-1\r\n"

                 这被称为Null Bulk String（空的大字符串）。  
  
                 当服务器使用空字符串进行回复时，客户端库API不应该返回空字符串，而是返回一个nil对象。例如，Ruby库应该返回'nil'，而C库应该返回NULL值（或者在应答对象中设置一个特殊的标志），等等。  
  
  
**8、RESP数组（RESP Arrays）**  
  
                 Redis客户端使用RESP数组发送命令到Redis服务器。同样，某些Redis命令使用 RESP数组 作为回复类型 将元素集合返回给客户端。一个例子是返回列表元素的LRANGE命令。  
  
                 RESP数组使用以下格式发送：  
  
                        1、一个 \* 字符作为第一个字节，后面跟着一个十进制的数字，该数字是数组中元素的个数，然后是CRLF。  
  
                        2、Array的每个元素都有一个额外的RESP类型。  
  
                  所以一个空的Array只是以下内容：

"\*0\r\n"

                   虽然两个RESP批量字符串“foo”和“bar”的数组编码为：

"\*2\r\n$3\r\nfoo\r\n$3\r\nbar\r\n"

                   正如您看到的那样， \* <count> CRLF 部分作为数组的前缀，组成数组的其他数据类型只是依次连接在一起。例如，一个三个整数的数组编码如下：

"\*3\r\n:1\r\n:2\r\n:3\r\n"

                   数组可以包含混合类型，元素之间不必是同一类型的。例如，一个四个整数和一个字符串块的列表可以编码如下：

[IMG_256](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

\*5\r\n

:1\r\n

:2\r\n

:3\r\n

:4\r\n

$6\r\n

foobar\r\n

[IMG_257](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                   （为了清楚起见，应答内容分为多行）。  
  
                    服务器发送的第一行是 \*5\r\n，以指定接下来的五个回复。然后，构成多批量回复的项目的每个回复都被传送。  
  
                    Null数组的概念也存在，并且是指定Null值的替代方法（通常使用Null Bulk String，但由于历史原因，我们有两种格式）。  
  
                    例如，当BLPOP命令超时时，它会返回一个空数组，其计数为-1，如下例所示：

"\*-1\r\n"

                   当Redis使用空数组响应时，客户端库API应返回空对象而不是空数组。这是区分空列表和不同条件（例如BLPOP命令的超时条件）所必需的。  
  
                   在RESP协议中也有可能存在数组的数组。例如，两个数组的数组编码如下：

[IMG_258](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

\*2\r\n

\*3\r\n

:1\r\n

:2\r\n

:3\r\n

\*2\r\n

+Foo\r\n

-Bar\r\n

[IMG_259](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                   （回复内容被分成多行，并加了空行，只是为了阅读方便）。  
  
                    上述RESP数据类型的编码表示了一个包含两个数组元素的数组，一个是包含三个整数1,2,3的一个数组，另一个是包含一个简单字符串和一个错误组成的两个元素的数组。  
  
  
**9、数组中的空元素（Null elements in Arrays）**  
  
                  数组中的单个元素可能为空。这用于Redis回复中，以表示这些元素缺失并且不是空的字符串。当SORT命令使用GET模式选项时，如果缺少指定的键，可能会发生这种情况。 包含Null元素的Array回复的示例：

[IMG_260](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

\*3\r\n

$3\r\n

foo\r\n

$-1\r\n

$3\r\n

bar\r\n

[IMG_261](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                 第二个元素是空值。 客户端库应该返回如下所示的内容：

["foo",nil,"bar"]

                   请注意，这不是前面章节中所述的异常情况，而只是进一步指定协议的一个示例。  
  
  
**10、将命令发送到Redis服务器（Sending commands to a Redis Server）**  
  
                    现在您已经熟悉RESP序列化格式，编写Redis客户端库的实现将很容易。我们可以进一步指定客户端和服务器之间的交互如何工作：  
  
                          1、客户端向Redis服务器发送仅包含Bulk Strings的RESP数组。  
  
                          2、Redis服务器回复发送任何有效的RESP数据类型作为客户端的回复。  
  
                    例如，一个典型的交互可能是如下这样。  
  
                    客户端发送命令 LLEN mylist以获取存储在键名为mylist中的列表的长度，并且服务器以如下例子回复一个整数应答（C：是客户端，S：服务器）。

[IMG_262](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

C: \*2\r\n

C: $4\r\n

C: LLEN\r\n

C: $6\r\n

C: mylist\r\n

S: :48293\r\n

[IMG_263](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                     通常我们将协议的不同部分用换行符分开，但实际的交互是客户端作为一个整体发送 \*2\r\n$4\r\nLLEN\r\n$6\r\nmylist\r\n。  
  
  
**11、多个命令和管道（Multiple commands and pipelining）**  
  
                     客户端可以使用相同的连接来发出多个命令。支持管道操作，因此客户端可以使用单个写入操作发送多个命令，而无需在发出下一条命令之前读取先前命令的服务器回复。所有的答复都可以在最后阅读。  
  
                      欲了解更多信息，请查看我们关于[管道的页面](https://redis.io/topics/pipelining" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)。  
  
  
**12、内联命令**  
  
                   有时在你的手中只有telnet工具，并且你需要发送一个命令到Redis服务器。虽然Redis协议易于实现，但在交互式会话中使用并不理想，而redis-cli可能并不总是可用。出于这个原因，Redis也以一种专门为人类设计的方式接受命令，并被称为内联命令格式。

C: PING

S: +PONG

                   以下是返回整数的内联命令的另一个示例：

C: EXISTS somekey

S: :0

                      基本上你只需在telnet会话中编写空格分隔的参数。由于没有以统一请求协议中使用的 \* 开始的命令，Redis 能够检测到这种情况并解析您的命令。  
  
  
**13、Redis协议的高性能分析器（High performance parser for the Redis protocol）**  
  
                     尽管Redis协议非常易于人工阅读并且易于实现，但它也可以通过类似二进制协议的性能来实现。  
  
                     RESP使用前缀长度来传输批量数据，因此永远不需要扫描特殊字符有效负载，例如使用JSON发生的情况，也不需要承担发送到服务器的有效负载。  
  
                     批量和多批量的长度可以使用代码进行计算，每个字符执行一次计算操作，同时扫描CR字符检查，像下面的C代码一样：

[IMG_264](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

#include <stdio.h>

int main(void) {

unsigned char \*p = "$123\r\n";

int len = 0;

p++;

while(\*p != '\r') {

len = (len\*10)+(\*p - '0');

p++;

}

/\* Now p points at '\r', and the len is in bulk\_len. \*/

printf("%d\n", len);

return 0;

}

[IMG_265](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                    在识别出第一个CR之后，可以在不进行任何处理的情况下将其与以下LF一起跳过。然后可以使用单个读取操作读取批量数据，该操作不会以任何方式检查有效负载。最后，剩余的 CR 和 LF 字符将被丢弃而不进行任何处理。  
  
                    虽然在性能上与二进制协议相当，但Redis协议在大多数高级语言中实现起来要简单得多，可减少在实现客户端的软件中的错误数量。  
  
  
**三、结束**  
        
            好了，今天就到这里了。该文章也翻译的差不多了，由于英文水平有限，翻译的过程中可能会出现一些语义不通的地方，希望大家指出，我可以修改错误，做的更好。好了，就说这么多吧，如果大家想查看原文，[请点击这里](https://redis.io/topics/protocol" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)。