Server Engineering Redis Challenge

Objective

Redis is an in-memory NoSQL data store that supports operations or “commands” on data structures such as sets, lists and hashes. Your objective is to implement a service that supports a subset of the Redis command set. That is, you are to build a “mini redis”.

Requirements

Command Requirements

You are to implement the following set of commands.

1. `SET key value`
2. `SET key value EX seconds` *(need not implement other SET options)*
3. `GET key`
4. `DEL key`
5. `DBSIZE`
6. `INCR key`
7. `ZADD key score member`
8. `ZCARD key`
9. `ZRANK key member`
10. `ZRANGE key start stop`

Atomicity

One of the key benefits of Redis is that it guarantees atomic, ordered access to data. Your server should offer the same guarantee.

Networking Requirements

When launched, it should listen on port 5555. You should **not** be concerned with following the Redis wire protocol; instead, we will make some radical simplifications:

1. Keys and values can only be the characters from the set [a-zA-Z0-9-_].
2. Commands are ASCII strings with space-delimited parameters.
3. Responses are ASCII strings with space-delimited values (where appropriate).

Other networking requirements:

1. Your server should support **multiple** simultaneous connections, just like real Redis

2. It should be possible to connect to the service via telnet. See example below.

Example

```
telnet localhost 5555
SET mykey cool-value
OK
GET mykey
cool-value
DEL mykey
OK
GET mykey
(nil)
```

References

[Redis command reference]

Library Support

In order to allow you to focus on implementing the core Redis features effectively, you should use a performant existing TCP server library to bootstrap your development.