Redis configuration

Redis is able to start without a configuration file using a built-in default configuration, however this setup is only recommended for testing and development purposes.

The proper way to configure Redis is by providing a Redis configuration file, usually called `redis.conf`.

The `redis.conf` file contains a number of directives that have a very simple format:

```
keyword argument1 argument2 ... argumentN
```

This is an example of configuration directive:

```
slaveof 127.0.0.1 6380
```

It is possible to provide strings containing spaces as arguments using (double or single) quotes, as in the following example:

```
requirepass "hello world"
```

Single-quoted string can contain characters escaped by backslashes, and double-quoted strings can additionally include any ASCII symbols encoded using backslashed hexadecimal notation "\xff".

The list of configuration directives, and their meaning and intended usage is available in the self documented example redis.conf shipped into the Redis distribution.

- The self documented redis.conf for Redis 6.0.
- The self documented redis.conf for Redis 5.0.
- The self documented redis.conf for Redis 4.0.
- The self documented redis.conf for Redis 3.2.
- The self documented redis.conf for Redis 3.0.
- The self documented redis.conf for Redis 2.8.
- The self documented redis.conf for Redis 2.6.
- The self documented redis.conf for Redis 2.4.

Passing arguments via the command line

Since Redis 2.6 it is possible to also pass Redis configuration parameters using the command line directly. This is very useful for testing purposes. The following is an example that starts a new Redis instance using port 6380 as a slave of the instance running at 127.0.0.1 port 6379.

```
./redis-server --port 6380 --slaveof 127.0.0.1 6379
```

The format of the arguments passed via the command line is exactly the same as the one used in the redis.conf file, with the exception that the keyword is prefixed with --.

Note that internally this generates an in-memory temporary config file (possibly concatenating the config file passed by the user if any) where arguments are translated into the format of redis.conf.

Changing Redis configuration while the server is running

It is possible to reconfigure Redis on the fly without stopping and restarting the service, or querying the current configuration programmatically using the special commands CONFIG SET and CONFIG GET

Not all the configuration directives are supported in this way, but most are supported as expected. Please refer to the CONFIG SET and CONFIG GET pages for more information.

Note that modifying the configuration on the fly **has no effects on the redis.conf file** so at the next restart of Redis the old configuration will be used instead.

Make sure to also modify the `redis.conf` file accordingly to the configuration you set using CONFIG SET. You can do it manually, or starting with Redis 2.8, you can just use CONFIG REWRITE, which will automatically scan your `redis.conf` file and update the fields which don't match the current configuration value. Fields non existing but set to the default value are not added. Comments inside your configuration file are retained.

Configuring Redis as a cache

If you plan to use Redis just as a cache where every key will have an expire set, you may consider using the following configuration instead (assuming a max memory limit of 2 megabytes as an example):

```
maxmemory 2mb
maxmemory-policy allkeys-lru
```

In this configuration there is no need for the application to set a time to live for keys using the EXPIRE command (or equivalent) since all the keys will be evicted using an approximated LRU algorithm as long as we hit the 2 megabyte memory limit.

Basically in this configuration Redis acts in a similar way to memcached. We have more extensive documentation about using Redis as an LRU cache.
