This README is just a fast *quick start* document. You can find more detailed documentation at http://redis.io.

What is Redis?

Redis is often referred as a *data structures* server. What this means is that Redis provides access to mutable data structures via a set of commands, which are sent using a *server-client* model with TCP sockets and a simple protocol. So different processes can query and modify the same data structures in a shared way.

Data structures implemented into Redis have a few special properties:

- Redis cares to store them on disk, even if they are always served and modified into the server memory. This means that Redis is fast, but that is also non-volatile.
- Implementation of data structures stress on memory efficiency, so data structures inside Redis will likely use less memory compared to the same data structure modeled using an high level programming language.
- Redis offers a number of features that are natural to find in a database, like replication, tunable levels of durability, cluster, high availability.

Another good example is to think of Redis as a more complex version of memcached, where the operations are not just SETs and GETs, but operations to work with complex data types like Lists, Sets, ordered data structures, and so forth.

If you want to know more, this is a list of selected starting points:

- Introduction to Redis data types. http://redis.io/topics/data-types-intro
- Try Redis directly inside your browser. http://try.redis.io
- The full list of Redis commands. http://redis.io/commands
- There is much more inside the Redis official documentation. http://redis.io/documentation

Building Redis

Redis can be compiled and used on Linux, OSX, OpenBSD, NetBSD, FreeBSD. We support big endian and little endian architectures, and both 32 bit and 64 bit systems.

It may compile on Solaris derived systems (for instance SmartOS) but our support for this platform is *best effort* and Redis is not guaranteed to work as well as in Linux, OSX, and *BSD there.

It is as simple as:

```
% make
```

RootFile

You can run a 32 bit Redis binary using:

```
% make 32bit
```

RootFile

After building Redis is a good idea to test it, using:

```
% make test
```

RootFile

Fixing build problems with dependencies or cached build options

Redis has some dependencies which are included into the `deps` directory. `make` does not rebuild dependencies automatically, even if something in the source code of dependencies is changes.

When you update the source code with `git pull` or when code inside the dependencies tree is modified in any other way, make sure to use the following command in order to really clean everything and rebuild from scratch:

```
make distclean
```

RootFile

This will clean: jemalloc, lua, hiredis, linenoise.

Also if you force certain build options like 32bit target, no C compiler optimizations (for debugging purposes), and other similar build time options, those options are cached indefinitely until you issue a `make distclean` command.

Fixing problems building 32 bit binaries

If after building Redis with a 32 bit target you need to rebuild it with a 64 bit target, or the other way around, you need to perform a `make distclean` in the root directory of the Redis distribution.

In case of build errors when trying to build a 32 bit binary of Redis, try the following steps:

- Install the packages libc6-dev-i386 (also try g++-multilib).
- Try using the following command line instead of `make 32bit` : `make CFLAGS="-m32 -march=native" LDFLAGS="-m32"`

Allocator

Selecting a non-default memory allocator when building Redis is done by setting the `MALLOC` environment variable. Redis is compiled and linked against libc malloc by default, with the exception of jemalloc being the default on Linux systems. This default was picked because jemalloc has proven to have fewer fragmentation problems than libc malloc.

To force compiling against libc malloc, use:

```
% make MALLOC=libc
```

RootFile

To compile against jemalloc on Mac OS X systems, use:

```
% make MALLOC=jemalloc
```

RootFile

Verbose build

Redis will build with a user friendly colorized output by default. If you want to see a more verbose output use the following:

```
% make V=1
```

RootFile

Running Redis

To run Redis with the default configuration just type:

```
% cd src
```

RootFile

```
% ./redis-server
```

If you want to provide your redis.conf, you have to run it using an additional parameter (the path of the configuration file):

```
% cd src
% ./redis-server /path/to/redis.conf
```

RootFile

It is possible to alter the Redis configuration passing parameters directly as options using the command line. Examples:

```
% ./redis-server --port 9999 --slaveof 127.0.0.1 6379
% ./redis-server /etc/redis/6379.conf --loglevel debug
```

RootFile referencing the cd

All the options in redis.conf are also supported as options using the command line, with exactly the same name.

Playing with Redis

You can use redis-cli to play with Redis. Start a redis-server instance, then in another terminal try the following:

```
% cd src
% ./redis-cli
redis> ping
PONG
redis> set foo bar
OK
redis> get foo
"bar"
redis> incr mycounter
(integer) 1
redis> incr mycounter
(integer) 2
redis>
```

RootFile

You can find the list of all the available commands at http://redis.io/commands.

Installing Redis

In order to install Redis binaries into /usr/local/bin just use:

RootFile

```
% make install
```

You can use `make PREFIX=/some/other/directory install` if you wish to use a different destination.

Make install will just install binaries in your system, but will not configure init scripts and configuration files in the appropriate place. This is not needed if you want just to play a bit with Redis, but if you are installing it the proper way for a production system, we have a script doing this for Ubuntu and Debian systems:

```
% cd utils
% ./install_server.sh
```

RootFile

The script will ask you a few questions and will setup everything you need to run Redis properly as a background daemon that will start again on system reboots.

You'll be able to stop and start Redis using the script named `/etc/init.d/redis_<portnumber>` , for instance `/etc/init.d/redis_6379` .

Code contributions

Note: by contributing code to the Redis project in any form, including sending a pull request via Github, a code fragment or patch via private email or public discussion groups, you agree to release your code under the terms of the BSD license that you can find in the COPYING file included in the Redis source distribution.

Please see the CONTRIBUTING file in this source distribution for more information.

Enjoy!