Redis Signals Handling

This document provides information about how Redis reacts to the reception of different POSIX signals such as `SIGTERM`, `SIGSEGV` and so forth.

The information contained in this document is **only applicable to Redis version 2.6 or greater**.

Handling of SIGTERM and SIGINT

The `SIGTERM` and `SIGINT` signals tell Redis to shutdown gracefully. When this signal is received the server does not immediately exit as a result, but it schedules a shutdown very similar to the one performed when the SHUTDOWN command is called. The scheduled shutdown starts ASAP, specifically as long as the current command in execution terminates (if any), with a possible additional delay of 0.1 seconds or less.

In case the server is blocked by a Lua script that is taking too much time, if the script is killable with SCRIPT KILL the scheduled shutdown will be executed just after the script is killed, or if terminates spontaneously.

The Shutdown performed in this condition includes the following actions:

- If there is a background child saving the RDB file or performing an AOF rewrite, the child is killed.
- If the AOF is active, Redis calls the `fsync` system call on the AOF file descriptor in order to flush the buffers on disk.
- If Redis is configured to persist on disk using RDB files, a synchronous (blocking) save is performed. Since the save is performed in a synchronous way no additional memory is used.
- If the server is daemonized, the pid file is removed.
- If the Unix domain socket is enabled, it gets removed.
- The server exits with an exit code of zero.

In case the RDB file can't be saved, the shutdown fails, and the server continues to run in order to ensure no data loss. Since Redis 2.6.11 no further attempt to shut down will be made unless a new `SIGTERM` will be received or the SHUTDOWN command issued.

Handling of SIGSEGV, SIGBUS, SIGFPE and SIGILL

The following follow signals are handled as a Redis crash:

- SIGSEGV
- SIGBUS
- SIGFPE

- SIGILL

Once one of these signals is trapped, Redis aborts any current operation and performs the following actions:

- A bug report is produced on the log file. This includes a stack trace, dump of registers, and information about the state of clients.
- Since Redis 2.8 a fast memory test is performed as a first check of the reliability of the crashing system.
- If the server was daemonized, the pid file is removed.
- Finally the server unregisters its own signal handler for the received signal, and sends the same signal again to itself, in order to make sure that the default action is performed, for instance dumping the core on the file system.

What happens when a child process gets killed

When the child performing the Append Only File rewrite gets killed by a signal, Redis handles this as an error and discards the (probably partial or corrupted) AOF file. The rewrite will be re-triggered again later.

When the child performing an RDB save is killed Redis will handle the condition as a more severe error, because while the effect of a lack of AOF file rewrite is a the AOF file enlargement, the effect of failed RDB file creation is lack of durability.

As a result of the child producing the RDB file being killed by a signal, or when the child exits with an error (non zero exit code), Redis enters a special error condition where no further write command is accepted.

- Redis will continue to reply to read commands.
- Redis will reply to all write commands with a `MISCONFIG` error.

This error condition is cleared only once it will be possible to create an RDB file with success.

Killing the RDB file without triggering an error condition

However sometimes the user may want to kill the RDB saving child without generating an error. Since Redis version 2.6.10 this can be done using the special signal `SIGUSR1` that is handled in a special way: it kills the child process as any other signal, but the parent process will not detect this as a critical error and will continue to serve write requests as usually.