sd_event_exit

- Description
- Return Value
  - Errors
- History
- See Also

systemd

sd_event_exit

3

sd_event_exit

sd_event_get_exit_code

Ask the event loop to exit

#include <systemd/sd-event.h>

int

sd_event_exit

sd_event *

event

int

code

int

sd_event_get_exit_code

sd_event *

event

int *

ret

Description

`sd_event_exit()` requests the event loop specified in the `event` event loop object to exit. The `code` parameter may be any integer value and is returned as-is by `sd_event_loop(3)` after the last event loop iteration. It may also be queried using `sd_event_get_exit_code()`, see below.

When exiting is requested the event loop will stop listening for and dispatching regular event sources. Instead it will proceed with executing only event sources registered with `sd_event_add_exit(3)` in the order defined by their priority. After all exit event sources have been dispatched the event loop is terminated.

If `sd_event_exit()` is invoked a second time while the event loop is still processing exit event sources, the exit code stored in the event loop object is updated, but otherwise no further operation is executed.

`sd_event_get_exit_code()` may be used to query the exit code passed to an earlier call of `sd_event_exit()`. The return parameter `ret` may be set to `NULL`, in order to simply check if `sd_event_exit()` has been called before (as `sd_event_get_exit_code()` fails with `-ENODATA` if that's not the case, see below).

While the full positive and negative integer ranges may be used for the exit code, care should be taken not pick exit codes that conflict with regular exit codes returned by `sd_event_loop()`, if these exit codes shall be distinguishable.

Note that for most event source types passing the callback pointer as `NULL` in the respective constructor call (i.e. in `sd_event_add_time(3)`, `sd_event_add_signal(3)`, …) has the effect of `sd_event_exit()` being invoked once the event source triggers, with the specified userdata pointer cast to an integer as the exit code parameter. This is useful to automatically terminate an event loop after some condition, such as a timeout or reception of `SIGTERM` or similar. See the documentation for the respective constructor call for details.

Return Value

On success, `sd_event_exit()` and `sd_event_get_exit_code()` return 0 or a positive integer. On failure, they return a negative errno-style error code.

Errors

Returned errors may indicate the following problems:

`-EINVAL`
The event loop object or error code pointer are invalid.

`-ECHILD`
The event loop was created in a different process, library or module instance.

`-ESTALE`
The event loop has exited already and all exit handlers are already processed.

`-ENODATA`
Returned by `sd_event_get_exit_code()` in case the event loop has not been requested to exit yet.

History

`sd_event_exit()` and `sd_event_get_exit_code()` were added in version 229.

See Also

`systemd(1)`, `sd-event(3)`, `sd_event_new(3)`, `sd_event_add_exit(3)`, `sd_event_add_time(3)`, `sd_event_add_signal(3)`, `sd_event_add_io(3)`, `sd_event_add_defer(3)`, `sd_event_add_inotify(3)`

□□□ #2
□ □□□ □□□ 2026-10-31 14:55:25 UTC
□ □□□ □□□ 2026-10-31 14:55:26 UTC