LiveRecorder 9.0 quick reference

Full documentation at docs.undo.io • Need help? Contact support@undo.io

LAUNCHING OR ATTACHING

`$ live-record PROGRAM [ARGUMENTS]`
Start PROGRAM under recording

`$ live-record --pid PID` or **`-p PID`**
Attach to process PID and start recording

`$ live-record --retry-for TIME PROGRAM`
Start PROGRAM repeatedly for TIME, until a recording is made (interacts with the **`--save-on`** option)

`--quiet` or **`-q`**
Suppress all LiveRecorder output apart from errors

WHEN TO START RECORDING

When attaching, live-record always starts recording immediately

`--record-on entry`
Start recording at the program entry point (default)

`--record-on immediate`
Start recording at the very first instruction, including the linker

`--record-on symbol:SYMBOL`
Start recording when execution first reaches SYMBOL

`--record-on program:PATTERN`
Record any child process whose name matches PATTERN

WHEN TO SAVE A RECORDING FILE

Save a recording only in specified circumstances, to allow recording of unreliable programs, only saving a recording on failure

`--save-on always`
Always save a recording at termination or detach (default)

`--save-on error`
Save a recording if the program terminates with an error

`--save-on exit-non-zero`
Save a recording if the program exits with a non-zero exit status

`--save-on exit-signal`
Save a recording if the program exits due to an unhandled signal

`--save-on exit-zero`
Save a recording if the program exits with a zero exit status

`--save-on detach`
Save a recording when live-record is detached from the program

`--save-on unsupported`
Save a recording if the program does something unsupported

`--save-on event-log-full`
Save a recording when the Undo event log is full

`--save-on A,B`
Save a recording if either condition A or B is satisfied

OUTPUT LOCATION

`--recording-file FILENAME` or **`-o FILENAME`**
Save any recording to FILENAME

`--recording-dir DIRNAME`
Save any recordings to DIRNAME

THREAD FUZZING

`--thread-fuzzing`
Enable thread fuzzing – vary thread scheduling to provoke bugs

`--thread-fuzzing-analyze REFERENCE` (experimental)
Use Undo recording REFERENCE to guide thread fuzzing;
The reference recording should be made with **`--disable-aslr`**

`--thread-fuzzing-feedback` (experimental)
Analyze execution while recording to guide thread fuzzing

CONTROLLING WITH SIGNALS

`live-record --list` or **`-l`**
List all currently running LiveRecorder sessions

Send `SIGINT`
Detach from the target process, saving according to **`--save-on`**

Send `SIGUSR1`
Detach from the target process, **always** saving a recording

Send `SIGTERM`
Detach from the target process, **never** saving a recording

EXTRA OPTIONS

`--disable-aslr`
Disable address space layout randomization for repeatability

`--max-event-log-size SIZE[K|M|G]`
Set the maximum size for the Undo event log

`--tmpdir-root DIRNAME`
Use DIRNAME for temporary file storage

`--verbose`
Print extra information at startup and when saving a recording

THE UNDOLR API

As well as the command-line live-record tool, Undo recordings can also be made by integrating the LiveRecorder API into your application.

This gives complete control over when to start and stop recording, and whether to save a recording file. For example:

```
#include "undolr.h"

int main(int argc, char *argv[]) {
    init_stage();

    undolr_start(...);
    critical_stage(...);
    undolr_save(FILENAME);
    undolr_stop(...);

    cleanup_stage();
}
```