Trace: Instrumentation to Show Function Calls

Version 9.4.0.3

November 1, 2026

Calltrace is a tool that displays all calls to user procedures. It displays the arguments to the calls, and it indents to show the depth of the continuation.

1 Quick Instructions

- Throw away `".zo"` versions of your source
- Prefix your program with

  ```
  (require trace)
  ```

  perhaps by starting `racket` with

  ```
  racket -l trace ...
  ```

  before arguments to load your program.
- Run your program

The `trace` module is odd; don't import it into another module. Instead, the `trace` module is meant to be invoked from the top-level, so that it can install an evaluation handler, exception handler, etc.

To reuse parts of the code of `trace`, import `trace/calltrace-lib`. It contains all of the bindings described here, but it but does not set the `current-eval` parameter.

2 Installing Calltrace

`(require trace)` package: `trace-lib`

Invoking the `trace` module sets the evaluation handler (via `current-eval`) to instrument Racket source code.

NOTE: `trace` has no effect on code loaded as compiled byte code (i.e., from a `".zo"` file) or native code (i.e., from a `".dll"`, `".so"`, or `".dylib"` file).

Calltrace's instrumentation can be explicitly disabled via the `instrumenting-enabled` parameter. Instrumentation is on by default. The `instrumenting-enabled` parameter affects only the way that source code is compiled, not the way that exception information is reported.

Do not load `trace` before writing `".zo"` files. Calltrace instruments S-expressions with unprintable values; this works fine if the instrumented S-expression is passed to the default eval handler, but neither the S-expression nor its byte-code form can be marshalled to a string.

3 Calltrace Library

```
(require trace/calltrace-lib)
```
package: trace-lib

The trace/calltrace-lib module provides functions that implement trace.

```
(instrumenting-enabled) → boolean?
(instrumenting-enabled on?) → void?
  on? : any/c
```

A parameter that determines whether tracing instrumentation is enabled.

```
(calltrace-eval-handler e) → any
  e : any/c
```

A procedure suitable for use with current-eval, which instruments expressions for Calltrace output (when instrumentation is not disabled via instrumenting-enabled).

Requiring trace installs this procedure as the value for current-eval.

```
(annotate e) → syntax?
  e : any/c
```

Instruments the expression e with Calltrace support. This function is normally used by calltrace-eval-handler.