Package 'winch'

October 5, 2020

Title Portable Native and Joint Stack Traces

Version 0.0.1

Date 2020-09-26

Description Obtain the native stack trace and fuse it with R's
stack trace for easier debugging of R packages with native code.

License GPL-3

URL https://r-prof.github.io/winch/, https://github.com/r-prof/winch

BugReports https://github.com/r-prof/winch/issues

Imports procmaps (>= 0.0.2)

Suggests DBI, knitr, magrittr, purrr, rlang, rmarkdown, RSQLite,
testthat, vctrs

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1.9000

NeedsCompilation yes

Author Kirill Müller [aut, cre] (<https://orcid.org/0000-0002-1416-3412>),
R Consortium [fnd],
Ian Lance Taylor [aut] (Bundled libbacktrace library),
Free Software Foundation [cph] (Bundled libbacktrace library)

Maintainer Kirill Müller <krlmlr+r@mailbox.org>

Repository CRAN

Date/Publication 2020-10-05 11:50:02 UTC

R topics documented:

`winch_add_trace_back` *Enrich an rlang traceback with details on native calls*

Description

This function uses the native stack trace returned from `winch_trace_back()` to add details on native function calls to an rlang traceback object. It is intended to be called by rlang.

Usage

```
winch_add_trace_back(trace = rlang::trace_back(bottom = parent.frame()))
```

Arguments

`trace`	An rlang traceback as returned by `rlang::trace_back()`.

Examples

```
foo <- function() {
  winch_call(bar)
}

bar <- function() {
  trace <- rlang::trace_back()
  winch_add_trace_back(trace)
}

foo()
```

`winch_available` *Are native tracebacks available?*

Description

Returns `TRUE` if `winch_trace_back()` is supported on this platform.

Usage

```
winch_available()
```

Value

A scalar logical.

Examples

```
winch_available()
```

`winch_call` *Call an R function from native code*

Description

Primarily intended for testing.

Usage

```
winch_call(fun, env = parent.frame())
```

Arguments

`fun`	A function callable without arguments.
`env`	The environment in which to evaluate the function call.

Value

The return value of `fun()`.

See Also

`winch_stop()`

Examples

```
foo <- function() {
  winch_call(bar)
}

bar <- function() {
  writeLines("Hi!")
}

foo()
```

winch_init_library *Set library to collect symbols for native stack traces*

Description

On Windows, function names in native stack traces can be obtained for only one library at a time. Call this function to set the library for which to obtain symbols.

Usage

```
winch_init_library(path = NULL, force = FALSE)
```

Arguments

path	Path to the DLL.
force	Reinitialize even if the path to the DLL is unchanged from the last call.

Value

This function is called for its side effects.

See Also

winch_call()

Examples

```
winch_init_library(getLoadedDLLs()[["rlang"]][["path"]])
```

winch_stop *Raise an error from native code*

Description

Primarily intended for testing.

Usage

```
winch_stop(message)
```

Arguments

message	The error message.

Value

This function throws an error and does not return.

See Also

`winch_call()`

Examples

```
try(winch_stop("Test"))
```

`winch_trace_back` *Native stack trace*

Description

This function returns the native stack trace as a data frame. Each native stack frame corresponds to one row in the returned data frame. Deep function calls come first, the last row corresponds to the running process's entry point.

Usage

```
winch_trace_back()
```

Details

On Windows, call `winch_init_library()` to return function names for a specific package.

Value

A data frame with the columns:

- `func`: function name
- `ip`: instruction pointer
- `pathname`: path to shared library

See Also

`sys.calls()` for the R equivalent.

Examples

```
winch_trace_back()

foo <- function() {
  winch_call(bar)
}

bar <- function() {
  winch_trace_back()
}

foo()
```

Index