Package: profmem (via r-universe)

August 24, 2024

Version 0.6.0

Title Simple Memory Profiling for R

Imports utils

Suggests R.rsp, markdown, microbenchmark

VignetteBuilder R.rsp

Description A simple and light-weight API for memory profiling of R expressions. The profiling is built on top of R's built-in memory profiler ('utils::Rprofmem()'), which records every memory allocation done by R (also native code).

License LGPL (>= 2.1)

LazyLoad TRUE

URL https://github.com/HenrikBengtsson/profmem

BugReports https://github.com/HenrikBengtsson/profmem/issues

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Repository https://henrikbengtsson.r-universe.dev

RemoteUrl https://github.com/HenrikBengtsson/profmem

RemoteRef master

RemoteSha ee3e0107156c8c0ef3209d34dbb4d5aa250c94d3

Contents

profmem *Memory profiling R*

Description

profmem() evaluates and memory profiles an R expression.

profmem_begin() starts the memory profiling of all the following R evaluations until profmem_end() is called.

Usage

```
profmem(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  threshold = getOption("profmem.threshold", 0L)
)

profmem_begin(threshold = getOption("profmem.threshold", 0L))

profmem_end()

profmem_suspend()

profmem_resume()

profmem_status()

profmem_depth()
```

Arguments

expr	An R expression to be evaluated and profiled.
envir	The environment in which the expression should be evaluated.
substitute	Should expr be base::substitute():d or not.
threshold	The smallest memory allocation (in bytes) to log.

Details

In order for memory profiling to work, R must have been *built* with memory profiling enabled. Function base::capabilities("profmem") will return TRUE of it is enabled, otherwise FALSE. If memory profiling is *not* supported, profmem() and profmem_begin() will produce an informative error. The pre-built R binaries on CRAN support memory profiling.

What is logged? The profmem() function uses utils::Rprofmem() for logging memory, which logs all memory *allocations* that are done via the R framework. Specifically, the logger is tied to allocVector3() part of R's native API. This means that nearly all memory allocations done in R

are logged. *Neither* memory deallocations *nor* garbage collection events are logged. Furthermore, allocations done by non-R native libraries or R packages that use native code `Calloc()` / `Free()` for internal objects are also *not* logged.

Any memory events that would occur due to calling any of the **profmem** functions themselves will *not* be logged and *not* be part of the returned profile data (regardless whether memory profiling is active or not). This is intentional.

If a profmem profiling is already active, `profmem()` and `profmem_begin()` performs an *independent*, *nested* profiling, which does not affect the already active one. When the active one completes, it will contain all memory events also collected by the nested profiling as if the nested one never occurred.

Profiling gathered by **profmem** *will* be corrupted if the code profiled calls `utils::Rprofmem()`, with the exception of such calls done via the **profmem** package itself.

Value

`profmem()` and `profmem_end()` returns the collected allocation data as an `Rprofmem` data.frame with additional attributes set. An `Rprofmem` data.frame has columns `what`, `bytes`, and `trace`, with:

- `what`: (character) type of memory event; either `"alloc"` or `"new page"`
- `bytes`: (numeric) number of bytes allocated or `NA_real_` (when `what` is `"new page"`)
- `trace`: (list of character vectors) zero or more function names

The attributes set are:

- `threshold` : The threshold used (= argument `threshold`)
- `expression`: The expression profiled (= argument `expr`)
- `value` : The value of the evaluated expression (only set if there was no error)
- `error` : The error object in case the evaluation failed (only set if there was an error)

`profmem_begin()` returns (invisibly) the number of nested profmem session currently active.

`profmem_suspend()` and `profmem_resume()` returns nothing.

`profmem_status()` returns `"inactive"`, `"active"`, or `"suspended"`.

`promem_depth()` returns a non-negative integer.

Examples

```
if (capabilities("profmem")) {

## Memory profile an R expression
p <- profmem({
  x <- raw(1000)
  A <- matrix(rnorm(100), ncol = 10)
})

## Display the results
print(p)

## Total amount of memory allocation
```

```
total(p)

## Allocations greater than 1 kB
p2 <- subset(p, bytes > 1000)
print(p2)

## The expression is evaluated in the calling environment
str(x)
str(A)

}
```

`readRprofmem` *Read an Rprofmem log file*

Description

Reads and parses an Rprofmem log file that was created by `utils::Rprofmem()`.

Usage

```
readRprofmem(pathname, as = c("Rprofmem", "fixed", "raw"), drop = 0L, ...)
```

Arguments

`pathname`	The Rprofmem log file to be read.
`as`	Specifies in what format data should be returned. If `"raw"`, the line content of the file is returned as is (as a character vector). If `"fixed"`, as `"raw"` but with missing newlines added to lines with empty stack calls that may be introduced in R (< 3.5.0) (see Ref. 1). If `"Rprofmem"`, the collected Rprofmem data is fully parsed into bytes and call stack information.
`drop`	Number of levels to drop from the top of the call stack.
`...`	Not used

Value

An `Rprofmem` data.frame or a character vector (if `as` is `"raw"` or `"fixed"`). An `Rprofmem` data.frame has columns `what`, `bytes`, and `trace`, with:

- `what`: (character) type of memory event; either `"alloc"` or `"new page"`
- `bytes`: (numeric) number of bytes allocated or `NA_real_` (when `what` is `"new page"`)
- `trace`: (list of character vectors) zero or more function names

References

Ref. 1: https://github.com/HenrikBengtsson/Wishlist-for-R/issues/25

Examples

```
file <- system.file("extdata", "example.Rprofmem.out", package = "profmem")

raw <- readRprofmem(file, as = "raw")
cat(raw, sep = "\n")

profmem <- readRprofmem(file, as = "Rprofmem")
print(profmem)
```

Index