Package ‘dotCall64’

October 7, 2016

Type Package

Title Enhanced Foreign Function Interface Supporting Long Vectors

Version 0.9-04

Date 2016-10-03

Author Kaspar Moesinger [aut, cre], Florian Gerber [ctb], Reinhard Furrer [ctb]

Maintainer Kaspar Moesinger <kaspar.moesinger@gmail.com>

Description An alternative version of .C() and .Fortran() supporting long vectors and 64-bit integer type arguments. The provided interface .C64() features mechanisms to avoid unnecessary copies of read-only or write-only arguments. This makes it a convenient and fast interface to C/C++ and Fortran code.

License GPL (>= 2)

URL https://git.math.uzh.ch/reinhard.furrer/dotCall64

BugReports https://git.math.uzh.ch/reinhard.furrer/dotCall64/issues

Depends R (>= 3.1)

Suggests microbenchmark, OpenMPController, RColorBrewer, roxygen2, spam, testthat,

Collate ‘vector_dc.R’ ‘dotCall64.R’ ‘zzz.R’

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-10-07 09:19:26

R topics documented:

  dotCall64 .......................................................... 2
  vector_dc ........................................................ 4

Index 6
Description

_C64_ can be used to call compiled and loaded C functions and Fortran subroutines. It works similar to _C_ and _Fortran_, and

1. supports long vectors, i.e., vectors with more than $2^{31}-1$ elements,
2. does the necessary castings to expose the R representation of "64-bit integers" (numeric vectors) to 64-bit integers arguments of the compiled function; int64_t types in C and integer (kind = 8) in Fortran,
3. provides a mechanism the control the duplication of the R objects exposed to the compiled code,
4. checks if the provided R objects are of the expected type and coerces the R object if necessary.

Compared to _C_, _C64_ has the additional arguments SIGNATURE, INTENT and VERBOSE. SIGNATURE specifies the types of the arguments of the compiled function. INTENT indicates whether the compiled function "reads", "writes", or "read and writes" the R objects passed to the compiled function. This is then used to duplicates R objects if (and only if) necessary.

Usage

\[
\text{C64}(.\text{NAME}, \text{SIGNATURE}, ..., \text{INTENT} = \text{NULL}, \text{NAOK} = \text{FALSE}, \text{PACKAGE} = "", \text{VERBOSE} = \text{getOption("dotCall64.verbose")})
\]

Arguments

\begin{itemize}
  \item \textbf{.NAME} a character vector of length 1. Specifies the name of the compiled function to be called.
  \item \textbf{SIGNATURE} a character vector of the same length as the number of arguments of the compiled function. Accepted strings are "double", "integer", "int64" describing the signature of each argument of the compiled function.
  \item \textbf{...} arguments passed to the compiled function. One R object for each argument. Up to 65 arguments are supported.
  \item \textbf{INTENT} a character vector of the same length as the number of arguments of the compiled code. Accepted strings are "rw", "r" or "w" indicating whether the intent of the argument is "read and write", "read", or "write", respectively. If the INTENT of an argument is "rw", the R object is copied and the compiled function receives a pointer to that copy. If the INTENT of an R object is "r", the compiled function receives a pointer to the R object itself. While this avoids copying and hence is more efficient in terms of speed and memory usage, it is absolutely necessary that the compiled function does not alter the object, since this corrupts the R object in the current R session. When the intent is "w", the corresponding input argument can be specified with the function \texttt{vector_dc} or its shortcuts \texttt{integer_dc} and \texttt{numeric_dc}. This avoids copying the passed R objects and
\end{itemize}
dotCall64

hence is more efficient in terms of speed and memory usage. By default, all arguments have intent "rw".

**NAOK**

logical vector of length 1. If FALSE (default), the presence of NA or NaN or Inf in the R objects passed through ... results in an error. If TRUE, any NA or NaN or Inf values in the arguments are passed on to the compiled function. The used time to check arguments (if FALSE) maybe considerable for large vectors.

**PACKAGE**

character vector of length 1. Specifies where to search for the function given in .NAME. This is intended to add safety for packages, which can use this argument to ensure that no other package can override their external symbols, and also speeds up the search.

**VERBOSE**

Numeric vector of length 1. If 0, no warnings are printed. If 1 warnings are printed (which may help to improve the performance of the call), if 2 additional debug information is given as warnings. The default value can be changed via the dotCall64.verbose option, which is set to 0 by default.

**Value**

A list similar to the ... list of arguments passed in (including any names given to the arguments), but reflecting any changes made by the compiled C or Fortran code.

**References**


**Examples**

```r
## Consider the following C function, which is included
## in the dotCall64 package:
## void get_c(double *input, int *index, double *output) {
##     output[0] = input[index[0] - 1];
## }
##
## # We can use .C64() the call it from R:
## .C64("get_c", SIGNATURE = c("double", "integer", "double"),
##     input = 1:10, index = 9, output = double(1))$output
##
## # Not run:
## # 'input' can be a long vector
## x_long <- double(2^31) # requires 16 GB RAM
## x_long[9] <- 9; x_long[2^31] <- -1
## .C64("get_c", SIGNATURE = c("double", "integer", "double"),
##     input = x_long, index = 9, output = double(1))$output
##
## # Since 'index' is of type 'signed int' resulting in a 32-bit integer,
## # it can only capture integers up to 2^31-1. To extend this,
## # we define the C function as follows:
```
## Description
Helper functions to be used in calls to `.C64`. The function `vector_dc` and its shortcuts `numeric_dc` and `integer_dc` return a R object of class c("vector_dc", "list") containing the necessary information (type and length) to allocate the vector (initialized with 0) inside the call to `.C64`. Using `vector_dc` together with `INTENT = "w"` argument of `.C64` leads to performance gains by avoiding unnecessary castings and copies.

## Usage
```
vector_dc(mode = "logical", length = 0L)
numeric_dc(length = 0)
integer_dc(length = 0)
```

vector_dc

Arguments

- **mode**  Character vector of length 1. Storage mode of the vector to allocate.
- **length**  Numeric vector of length 1. Length of the vector to allocate.

Value

Object of class `vector_dc` and `list`.

Examples

```r
vector_dc("integer", 20)
```
Index

.C, 2
.C64, 4
.C64 (dotCall64), 2
.Fortran, 2
dotCall64, 2
integer_dc, 2
integer_dc (vector_dc), 4
numeric_dc, 2
numeric_dc (vector_dc), 4
vector_dc, 2, 4