

Package ‘Rserve’

February 20, 2012

Version 0.6-8

Title Binary R server

Author Simon Urbanek <Simon.Urbaneck@r-project.org>

Maintainer Simon Urbanek <Simon.Urbaneck@r-project.org>

Depends R (>= 1.5.0)

SystemRequirements R must be compiled with --enable-R-shlib if the server is to be built

Description Rserve acts as a socket server (TCP/IP or local sockets) which allows binary requests to be sent to R. Every connection has a separate workspace and working directory. Client-side implementations are available for popular languages such as C/C++ and Java, allowing any application to use facilities of R without the need of linking to R code. Rserve supports remote connection, user authentication and file transfer. A simple R client is included in this package as well.

License GPL-2

URL <http://www.rforge.net/Rserve/>

Repository CRAN

Date/Publication 2012-02-20 20:06:18

R topics documented:

| | |
|-------------------|---|
| Rclient | 2 |
| Rserve | 4 |
| self | 5 |

| | |
|--------------|----------|
| Index | 7 |
|--------------|----------|

Description

Rserve is a server providing R functionality via sockets. The following functions allow another R session to start new Rserve sessions and evaluate commands. The support is very rudimentary and uses only a fraction of the functionality provided by Rserve. The typical use of Rserve is to connect to other applications, not necessarily to connect two R processes. However, it is not uncommon to have a cluster of Rserve machines so the following functions provide a simple client access.

For more complete client implementation see `src/clients` directory of the Rserve distribution which show a C/C++ client. Also available from the Rserve pages is a Java client (JRclient). See <http://rosuda.org/Rserve> for details.

Usage

```

RSconnect(host = "localhost", port = 6311)
RSlogin(c, user, pwd, silent = FALSE)
RSeval(c, expr)
RSeval.old(c, cmd)
RSclose(c)
RSshutdown(c, pwd = NULL, ctrl = FALSE)
RSdetach(c)
RSevalDetach(c, cmd = "")
RSattach(session)
RSassign(c, obj, name = deparse(substitute(obj)) )
RSassign.old(c, obj, name = deparse(substitute(obj)) )
RSserverEval(c, expr)
RSserverSource(c, file)

```

Arguments

| | |
|---------|--|
| host | host to connect to |
| port | TCP port to connect to |
| c | Rserve connection |
| user | username for authentication |
| pwd | password for authentication |
| cmd | command (as string) to evaluate |
| silent | flag indicating whether a failure should raise an error or not |
| session | session object as returned by RSdetach or RSevalDetach |
| obj | value to assign |
| name | name to assign to on the remote side |
| expr | R expression to evaluate remotely |

| | |
|------|---|
| file | path to a file on the server(!) that will be sourced into the main instance |
| ctrl | logical, if TRUE then control command (CMD_ctrlShutdown) is used for shutdown, otherwise the legacy CMD_shutdown is used instead. |

Details

RScconnect creates a connection to a Rserve. The returned handle is to be used in all subsequent calls to client functions. The session associated with the connection is alive until closed via RSclose.

RSlogin performs authentication with the Rserve. Currently this simple client supports only plain text authentication, encryption is not supported.

RSclose closes the Rserve connection.

RSeval evaluates the supplied expression remotely. `expr` can be either a string or any R expression. Use `quote` to use unevaluated expressions. The implementation of `RSeval` is very efficient in that it does not require any buffer on the remote side and uses native R serialization as the protocol. See examples below for correct use.

`RSeval.old` is deprecated and present only for compatibility with older Rserve implementations. It evaluates the string in the remote Rserve and returns the result. Note that you will have to load the same packages on both ends if the result is an (S3/S4) object such that corresponding classes and methods are available. Also note that the result is transported on the network so sending huge results can be slow. Thus consider sending only relevant parts or keep the results on the other end if pertinent.

RSdetach detaches from the current Rserve connection. The connection is closed but can be restored by using `RSattach` with the value returned by `RSdetach`. Technically the R on the other end is still running and waiting to be attached.

RSshutdown terminates the server gracefully. It should be immediately followed by `RSclose` since the server closes the connection. It can be issued only on a valid (authenticated) connection. The password parameter is currently ignored since password-protected shutdown is not yet supported. Please note that you should not terminate servers that you did not start. More recent Rserve installation can disable regular shutdown and only allow control shutdown (available to control users only) which is invoked by specifying `ctrl=TRUE`.

`RSevalDetach` same as `RSdetach` but allows asynchronous evaluation of the command. The remote Rserve is instructed to evaluate the command after the connection is detached. Please note that the session cannot be attached until the evaluation finished. Therefore it is advisable to use another session when attaching to verify the status of the detached session where necessary.

`RSattach` resume connection to an existing session in Rserve. The `session` argument must have been previously returned from the `RSdetach` or `RSevalDetach` comment.

`RSassign` pushes an object to Rserve and assigns it to the given name. Note that the name can be an (unevaluated) R expression itself thus allowing constructs such as `RSassign(c, 1:5, quote(a$foo))` which will result in `a$foo <- 1:5` remotely. However, character names are interpreted literally.

`RSserverEval` and `RSserverSource` enqueue commands in the server instance of Rserve, i.e. their effect will be visible for all subsequent client connections. The Rserve instance must have control commands enabled (not the default) in order to allow those commands. `RSserverEval` evaluates the supplied expression and `RSserverSource` sources the specified file - it must be a valid path to a file on the server, not the client machine! Both commands are executed asynchronously in the server, so the success of those commands only means that they were queued on the server - they

will be executed between subsequent client connections. Note that only subsequent connections will be affected, not the one issuing those commands.

Author(s)

Simon Urbanek

Examples

```
## Not run:
c <- RSconnect()
data(stackloss)
RSassign(c, stackloss)
RSeval(c, quote(library(MASS)))
RSeval(c, quote(rlm(stack.loss ~ ., stackloss)$coeff))
RSeval(c, "getwd()")

image <- RSeval(c, quote(try({
  attach(stackloss)
  library(Cairo)
  Cairo(file="plot.png")
  plot(Air.Flow, stack.loss, col=2, pch=19, cex=2)
  dev.off()
  readBin("plot.png", "raw", 999999)})))
if (inherits(image, "try-error"))
  stop(image)

## End(Not run)
```

Rserve

*Server providing R functionality to applications via TCP/IP or local
unix sockets*

Description

Starts Rserve in daemon mode (unix only). Any additional parameters not related to Rserve will be passed straight to the underlying R. For configuration, usage and command line parameters please consult the online documentation at <http://www.rforge.net/Rserve>. Use `R CMD Rserve --help` for a brief help.

The Rserve function is provided for convenience only, it is recommended to start Rserve directly from the command line, not from R itself. Also note that the debug version of Rserve doesn't fork and thus will block until closed.

On Windows the Rserve() function sets up the PATH to include the current R.DLL so that Rserve can be run.

Usage

```
# R CMD Rserve [<parameters>]
```

```
Rserve(debug = FALSE, port = 6311, args = NULL, quote=(length(args) > 1), wait, ...)
```

Usage

```
Rserve(debug = FALSE, port = 6311, args = NULL, quote=(length(args) > 1), wait, ...)
```

Arguments

| | |
|-------|---|
| debug | determines whether regular Rserve or debug version of Rserve (Rserve.dbg) should be started. |
| port | port used by Rserve to listen for connections |
| args | further arguments passed to Rserve (as a string that will be passed to the system command - see quote below). |
| quote | logical, if TRUE then arguments are quoted, otherwise they are just joined with spaces |
| wait | wait argument for the <code>system</code> call. It defaults to FALSE on Windows and TRUE elsewhere. |
| ... | other arguments to be passes to <code>system</code> . |

Details

Rserve is not just a package, but an application. It is provided as a R package for convenience only. For details see <http://www.rforge.net/Rserve>

Author(s)

Simon Urbanek

self

Functions usable for R code run inside Rserve

Description

The following functions can only be used inside Rserve, they cannot be used in stand-alone R. They interact with special features of Rserve. All commands below will succeed only if Rserve has been started with `r-control enable` configuration setting for security reasons.

`self.ctrlEval` issues a control command to the Rserve parent instance that evaluates the given expression in the server. The expression is only queued for evaluation which will happen asynchronously in the server (see [RSserverEval](#) for details). Note that the current session is unaffected by the command.

`self.ctrlSource` issues a control command to the Rserve parent instance to source the given file in the server, see [RSserverSource](#) for details.

Usage

```
self.ctrlEval(expr)
self.ctrlSource(file)
```

Arguments

`expr` R expression to evaluate remotely
`file` path to a file that will be sourced into the main instance

Value

Both functions return TRUE (invisibly).

Author(s)

Simon Urbanek

Examples

```
## Not run:  
  self.ctrlEval("a <- rnorm(10)")  
  
## End(Not run)
```

Index

*Topic **interface**

Rclient, [2](#)

Rserve, [4](#)

self, [5](#)

quote, [3](#)

Rclient, [2](#)

RSassign (Rclient), [2](#)

RSattach (Rclient), [2](#)

RSclose (Rclient), [2](#)

RSconnect (Rclient), [2](#)

RSdetach (Rclient), [2](#)

Rserve, [4](#)

RSeval (Rclient), [2](#)

RSevalDetach (Rclient), [2](#)

RSlogin (Rclient), [2](#)

RSserverEval, [5](#)

RSserverEval (Rclient), [2](#)

RSserverSource, [5](#)

RSserverSource (Rclient), [2](#)

RSshutdown (Rclient), [2](#)

self, [5](#)

system, [5](#)