Package 'orcv'

November 18, 2023

Type Package

Title Async Communicating Event Receiver

Version 1.1

Date 2022-10-14

Author Jason Cairns

Maintainer <jason.cairns@auckland.ac.nz>

Description Self-contained package that creates an event queue which can be responded to with long-running connections.

Suggests parallel

License MIT

NeedsCompilation yes

R topics documented:

receive 1
send 3
start 4

Index **5**

`receive` *Receive external* **orcv** *communications*

Description

If *x* is missing, blocking pops the oldest Message object from the communication queue as created by `start`. An open connection with the sender may be maintained if *keep_conn* is set to `TRUE`. Alternatively, if an existing connection as given by a `FD` object is passed as *x*, then the file descriptor is listened on. S3 method dispatching on *x*.

Usage

```
receive(x, keep_conn = FALSE, simplify = TRUE, ...)
receive.FD(x, keep_conn=FALSE, simplify=TRUE,...)
```

Arguments

x	Optional file descriptor or other S3 class. Reads from background communication queue if missing.
keep_conn	Logical whether to maintain or close the connection.
simplify	Simplify Message output.
...	Further arguments to methods.

Value

If from the message queue, a Message, composed of components:

header

payload

location

fd

Arbitrary otherwise.

See Also

send, start

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, keep_conn = FALSE, simplify = TRUE, ...)
{
    stopifnot(ORCV_GLOBAL$STARTED)
    if (missing(x)) {
        next_msg <- .Call(C_next_message)
        if (is.null(next_msg))
            stop("receive error")
        msg <- as.Message(next_msg)
        cat(sprintf("Opening message with header \"%s\"\n", header(msg)))
        if (!keep_conn) {
            close(msg)
            fd(msg) <- as.FD(-1L)
        }
        if (!simplify)
            msg <- list(msg)
        invisible(msg)
    }
    else UseMethod("receive", x)
  }
```

send *Send an R object to a remote location*

Description

S3 generic to send an R object to a location that may be method-defined. Message objects encapsulate a payload along with an address, but other classes require a destination as the *x* argument, either through a FD class, or as a character hostname.

Usage

```
send(x, ...)
send.Message(x, header, payload=NULL, keep_conn=FALSE, ...)
send.FD(x, header, payload=NULL, keep_conn=FALSE, ...)
send.character(x, port, header, payload=NULL, keep_conn=FALSE, ...)
```

Arguments

x	Destination, or object encapsulating one, serving as S3 dispatch.
...	Passed on to further methods.

Value

Typically a File Descriptor of the connection if successful. -1L if connection is not kept.

See Also

send, start

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, ...)
{
    stopifnot(ORCV_GLOBAL$STARTED)
    stopifnot(length(x) > 0)
    UseMethod("send", x)
  }
```

start *Begin Communication Node*

Description

This function starts a communication node at the machine on which it is run. The communication node exists as a message queue running on a separate thread. Received messages are stored in a local queue and may be retrieved via the `receive` function.

Usage

```
start(address = NULL, port = 0L, threads = getOption("orcv.cores", 4L))
```

Arguments

`address`	A character address for the communication node to be reachable by. Leave `NULL` for localhost.
`port`	Integer port to bind to.
`threads`	Number of threads made available to the listening queue. Controlled by the `"orcv.cores"` option.

Value

An invisible `0L` if no error.

See Also

`receive`, `send`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (address = NULL, port = 0L, threads = getOption("orcv.cores",
    4L))
{
    stopifnot(is.character(address) || is.null(address))
    res <- .Call(C_start, address, as.integer(port), as.integer(threads))
    ORCV_GLOBAL$STARTED <- TRUE
    invisible(res)
  }
```

Index

∗ **programming**
`start`, 4

`receive`, 1, *4*

`send`, *2*, *3*, 3, *4*
`start`, *1*–*3*, 4