Package 'mori'

June 9, 2026

Title Shared Memory for R Objects

Version 0.2.1

Description Share R objects across processes on the same machine via a single copy in 'POSIX' shared memory (Linux, macOS) or a 'Win32' file mapping (Windows). Every process reads from the same physical pages through the R Alternative Representation ('ALTREP') framework, giving lazy, zero-copy access. Shared objects serialize compactly as their shared memory name rather than their full contents.

License MIT + file LICENSE

URL https://shikokuchuo.net/mori/, https://github.com/shikokuchuo/mori

BugReports https://github.com/shikokuchuo/mori/issues

Depends R (>= 4.3)

Suggests lobstr, mirai, testthat (>= 3.0.0)

Config/build/compilation-database true

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation yes

Author Charlie Gao [aut, cre] (ORCID: <https://orcid.org/0000-0002-0750-061X>), Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

Maintainer Charlie Gao <charlie.gao@posit.co>

Repository CRAN

Date/Publication 2026-06-09 08:30:02 UTC

Contents

mori-package 2
is_shared 3

map_shared 3
prune_shared 4
share 5
shared_name 6

Index **8**

`mori-package` *mori: Shared Memory for R Objects*

Description

Share R objects via shared memory with `share()`, access them in other processes with `map_shared()`, using R's ALTREP framework for zero-copy memory-mapped access. Shared objects serialize compactly via ALTREP serialization hooks. Shared memory is automatically freed when the R object is garbage collected.

Author(s)

Maintainer: Charlie Gao <charlie.gao@posit.co> (ORCID)

Authors:

- Charlie Gao <charlie.gao@posit.co> (ORCID)

Other contributors:

- Posit Software, PBC (ROR) [copyright holder, funder]

See Also

Useful links:

- https://shikokuchuo.net/mori/
- https://github.com/shikokuchuo/mori
- Report bugs at https://github.com/shikokuchuo/mori/issues

is_shared *Test if an Object is Shared*

Description

Returns TRUE if x is an ALTREP object backed by shared memory (created by share() or map_shared()), FALSE otherwise.

Usage

```
is_shared(x)
```

Arguments

x	an R object.

Value

TRUE or FALSE.

Examples

```
x <- 1:100
y <- share(x)
is_shared(y)
is_shared(x)
```

map_shared *Open Shared Memory by Name*

Description

Open a shared memory region identified by a name string and return an ALTREP-backed R object that reads directly from shared memory.

Usage

```
map_shared(name)
```

Arguments

name	a character string as returned by shared_name(): either a bare shared memory name (opens the root) or a name with a 1-based bracketed index path (e.g. "/mori_abc_1[2,3]", opens the addressed sub-list or element directly).

Value

The R object stored at the named region (or sub-object at the given path), or `NULL` if `name` is not a valid shared memory name (wrong type, length, `NA`, missing or malformed prefix, or malformed bracketed path). If `name` parses as valid but the region is absent or corrupted — or the path doesn't address a valid sub-object — an error is raised.

See Also

`share()` to create a shared object, `shared_name()` to extract the shared memory name.

Examples

```
x <- share(1:100)
nm <- shared_name(x)
map_shared(nm)

# A bracketed index path opens the addressed sub-object directly:
lst <- share(list(a = 1:3, b = letters))
map_shared(shared_name(lst[[2]]))
```

---

prune_shared *Prune Orphaned Shared Memory Regions*

Description

Recover shared memory regions leaked by a process that was killed before it could clean up — for example after a crash, a `SIGKILL`, or the out-of-memory killer. **You do not normally need this:** mori frees shared memory automatically when the `share()` object is garbage-collected and on a clean R session exit (see Details).

This function is only relevant on Linux and macOS. On Windows, shared memory cannot be orphaned, so there is never anything to clean up and calling it has no effect.

Usage

```
prune_shared()
```

Details

Shared memory is normally managed automatically: a region is unlinked when the `share()` object that owns it is garbage-collected, and on a clean R session exit. A region is only left behind if the owning process is killed before either can run. `prune_shared()` removes such leftovers.

Pruning is **conservative**: a region is removed only if its creating process (encoded in the region name) is no longer running, so regions still in use by a live process are never touched. Removing a region unlinks only its name; processes that have already mapped it keep reading until they release it, and the memory is freed once the last mapping is gone.

Value

Invisibly, a character vector of the region names that were removed, or NULL if nothing was removed.

Examples

```
# Shared memory is freed automatically - you do not normally call this.
# To prune regions left behind by a crashed process:
prune_shared()
```

---

share *Create a Shared Object*

Description

Write an R object into shared memory and return a version that other processes on the same machine can map without copying.

Usage

```
share(x)
```

Arguments

x	an R object.

Details

Attributes are stored alongside the data in the shared memory region and restored on the consumer side. Character vectors use a packed layout and elements are materialised lazily on access. When serialised (e.g. by `serialize()` or across a `mirai()` call), a shared object is represented compactly by its shared memory name (~30 bytes) rather than by its contents.

The shared memory region is managed automatically. It stays alive as long as the returned object (or any element extracted from it) is referenced in R, and is freed automatically when no references remain or the session exits cleanly.

`share()` is idempotent: calling it on an object that is already backed by shared memory returns the input unchanged without allocating a new region.

Important: ensure the return value of `share()` is not garbage collected before a consumer can map its shared memory.

Value

For atomic vectors (including character vectors and those with attributes such as names, dim, class, or levels) and lists or data frames whose elements are such vectors, an ALTREP-backed object that reads directly from shared memory. For any other object (environments, closures, language objects, NULL), the input is returned unchanged with no shared memory region created.

Persistence

Direct `saveRDS()` of a shared object writes only the shared memory name, so the resulting file is meaningful only on the same machine while the region is still alive. For portable storage or transport across machines, materialise into a regular in-memory copy first with `rlang::duplicate()`, which deep-duplicates the object:

```
saveRDS(rlang::duplicate(x), file = "x.rds")
```

See Also

`map_shared()` to open a shared region by name, `shared_name()` to extract the shared memory name.

Examples

```
x <- share(1:100)
sum(x)

lst <- share(list(a = 1:3, b = letters))
is_shared(lst)
```

---

shared_name *Extract Shared Memory Name*

Description

Extract the shared memory name from a shared object. This name can be passed to `map_shared()` to open the same region in another process.

Usage

```
shared_name(x)
```

Arguments

`x`	a shared object as returned by `share()` or `map_shared()`.

Value

A character string identifying the shared object, or `NULL` if `x` is not a shared object. For a sub-list or element extracted from a shared list, the string carries a bracketed 1-based index path (e.g. `"/mori_abc_1[2,3]"`). `map_shared()` accepts both forms; the path-qualified form returns the addressed sub-object directly. The underlying shared memory region name is the prefix before `[` and is recoverable via `sub("\\[.*$", "", shared_name(x))`.

See Also

`map_shared()` to open a shared region by name.

Examples

```
x <- share(1:100)
shared_name(x)

# A sub-object extracted from a shared list carries a bracketed index path:
lst <- share(list(a = 1:3, b = letters))
shared_name(lst[[2]])
```

Index

is_shared, 3

map_shared, 3
map_shared(), *2*, *3*, *6*
mori *(*mori-package*)*, 2
mori-package, 2

prune_shared, 4

saveRDS(), *6*
serialize(), *5*
share, 5
share(), *2–4*, *6*
shared_name, 6
shared_name(), *3*, *4*, *6*