# Package: mori (via r-universe)

June 5, 2026

**Title** Shared Memory for R Objects

**Version** 0.2.0.9000

**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

**Repository** https://shikokuchuo.r-universe.dev

**Date/Publication** 2026-06-05 21:52:16 UTC

**RemoteUrl** https://github.com/shikokuchuo/mori

**RemoteRef** HEAD

**RemoteSha** 3014cfb98beb6d684ad6821e4e4671b1427285bc

## Contents

---

`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 <- share(rnorm(100))
is_shared(x)
is_shared(rnorm(100))
```

## 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)
y <- map_shared(nm)
sum(y)
```

---

# 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 by the garbage collector when no references remain.

`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**: always assign the result of `share()` to a variable. The shared memory is kept alive by the R object reference — if the result is used as a temporary (not assigned), the garbage collector may free the shared memory before a consumer process has mapped it.

## 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(rnorm(100))
sum(x)
```

---

# 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(rnorm(100))
shared_name(x)
```

---

`unlink_shared` *Unlink Shared Memory Regions*

---

## Description

Remove shared memory regions created by `share()`. Pass one or more region names to remove specific regions, or call with no arguments to reap *orphaned* regions: those whose creating process has died without cleaning up, for example after a crash, a `SIGKILL`, or the out-of-memory killer.

## Usage

```
unlink_shared(name = NULL)
```

## Arguments

| | |
|---|---|
| `name` | a character vector of shared memory names as returned by `shared_name()`, or `NULL` (the default) to reap orphaned regions. A sub-object path (e.g. `"/mori_abc_1[2,3]"`) is accepted and resolves to its underlying region. Names that are not valid mori identifiers are ignored. |

## 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. `unlink_shared()` removes such leftovers.

Unlinking a region removes only its name. Processes that have already mapped it keep reading until they release it; the memory is freed once the last mapping is gone.

The reap form (`name = NULL`) 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.

## Value

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

## Platform behaviour

**Linux** Both forms are supported; reaping enumerates `/dev/shm`.

**macOS** Removal by name is supported. Reaping is not: the kernel shared memory namespace cannot be enumerated, so names must be passed explicitly.

**Windows** Nothing is removed. A file mapping is reference-counted by the operating system and cannot be orphaned, so there is no leftover to clean up.

## See Also

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

## Examples

```
x <- share(rnorm(100))
nm <- shared_name(x)
rm(x)
unlink_shared(nm)
```

# Index