Package: cxxfunplus (via r-universe)

October 16, 2024

Type Package

Title extend cxxfunction by saving the dynamic shared objects

Version 1.0.2

Date 2012-09-6

Depends inline

Imports methods

Suggests Rcpp (>= 0.8.0)

Author Jiqiang Guo <guojq28@gmail.com>

Maintainer Jiqiang Guo <guojq28@gmail.com>

Description extend cxxfunction by saving the dynamic shared objects
for reusing across R sessions

License GPL-3

URL https://github.com/maverickg/cxxfunplus

Repository https://maverickg.r-universe.dev

RemoteUrl https://github.com/maverickg/cxxfunplus

RemoteRef HEAD

RemoteSha 8f77a3cdd14d3ce375dec11a826aeba3e6cd7146

Contents

`cxxfunplus-package` *cxxfunplus: save the dynamic shared objects (DSO) for cxxfunction*

Description

The cxxfunction function in **inline** could not save the dynamic shared objects (DSO) created in a session. We provide a mechanism to save the DSO's if for example, `save.image` is called.

Details

Instead of calling cxxfunction in **inline**, call `cxxfunctionplus` in this package, from which an S4 class of `cxxdso` is returned. We could use generic function `grab.cxxfun` of class `cxxdso` to retrieve the functions typically returned by `cxxfunction`.

Author(s)

Jiqiang Guo <guojq28@gmail.com>

Maintainer: Jiqiang Guo <guojq28@gmail.com>

See Also

`cxxfunctionplus`, `inline`

`cxxdso-class` *Class* `"cxxdso"`

Description

An S4 class for saving the dynamic shared objects created on the fly

Objects from the Class

Objects can be created by calls of `cxxfunctionplus`.

Slots

`sig`**:** Object of class `"list"` The signatures of functions defined.

`dso_saved`**:** Object of class `"logical"` Whether to save the DSO or not.

`dso_filename`**:** Object of class `"character"` The original file name for the DSO when it is created (no extension).

`system`**:** The operating system where the object is created.

`.MISC`**:** Object of class `"environment"` An environment to save the functions returned by `cxxfunction` with name `cxxfun`, the last full path for the DSO with name `dso_last_path`, and the vector of `raw` for saving the binary dynamic shared object (DSO) with name `dso_bin`.

Methods

show `signature(x = "cxxdso")`: Print a summary of the object.

grab_cxxfun `signature(object = "cxxdso")`: Return the function objects contained.

is_dso_loaded `signature(object = "cxxdso")`: Tell if the DSO (DLL) is loaded.

getDynLib `signature(x = "cxxdso")`: Obtain the DLL associated.

See Also

`getDynLib`, `grab_cxxfun`, and `cxxfunctionplus`

Examples

```
showClass("cxxdso")
```

`cxxfunctionplus` *To created an S4 class* `cxxdso` *from C++ code*

Description

This is a wrap-up of function `cxxfunction` in package **inline**. Additionally, this function returns an object of class `cxxdso`, which could be saved and reused across R sessions. All arguments except `save_dso` are passed to function `cxxfunction`.

Usage

```
cxxfunctionplus(sig = character(), body = character(),
                plugin = "default", includes = "",
                settings = getPlugin(plugin),
                save_dso = FALSE, ..., verbose = FALSE)
```

Arguments

`sig`	Signature of the function. A named character vector.
`body`	A character vector with C++ code to include in the body of the compiled C++ function.
`plugin`	Name of the plugin to use. See `getPlugin` for details about plugins.
`includes`	User includes, inserted after the includes provided by the plugin.
`settings`	Result of the call to the plugin.
`save_dso`	Determine whether to save the compiled code (DSO); defaults to `FALSE`.
`...`	Further arguments to the plugin.
`verbose`	verbose output.

Value

An object of S4 class `cxxdso`.

See Also

cxxfunction and cxxdso

Examples

```
## Not run:
src <-  ' return ScalarReal(INTEGER(x)[0] * REAL(y)[0]);'
dso <- cxxfunctionplus(signature(x = "integer", y = "numeric"), src)
show(dso)

## End(Not run)
```

getDynLib-methods   *Retrieve the dynamic library (or DLL) associated with an object of class* cxxdso

Description

The getDynLib function retrieves the dynamic library (or DLL) associated with objects of class cxxdso generated by cxxfunctionplus

Methods

signature(x = "cxxdso") Retrieves the dynamic library associated with the cxxdso objects generated by cxxfunctionplus.

See Also

getLoadedDLLs, dyn.load, cxxdso, and getDynLib in **inline**

Examples

```
## Not run:
dso <- cxxfunctionplus(signature(), "return R_NilValue;")
dll <- getDynLib(dso)

## End(Not run)
```

grab_cxxfun-methods *Retrieve the functions object associated with an object of class* cxxdso

Description

The grab_cxxfun function retrieves the function object associated with objects of class cxxdso generated by cxxfunctionplus

Methods

signature(x = "cxxdso") Retrieves the function object associated with the cxxdso objects generated by cxxfunctionplus.

See Also

cxxfunctionplus, cxxdso

Examples

```
## Not run:
dso <- cxxfunctionplus(signature(), "return R_NilValue;")
fx <- grab_cxxfun(dso)
fx()

## End(Not run)
```

is_dso_loaded-methods *Tell if a* cxxdso *object is loaded*

Description

The is_dso_loaded function tell if the dynamic shared object (DSO, or DLL) in an object of cxxdso, created by function cxxfunctionplus, is loaded.

Methods

signature(x = "cxxdso") Tell if a cxxdso object is loaded in the sense that the contained DSO is loaded or not.

See Also

cxxdso

Examples

```
## Not run:
dso <- cxxfunctionplus(signature(), "return R_NilValue ;")
print(is_dso_loaded(dso))

## End(Not run)
```

`is_null_cxxfun` *Tell if the address of functions created by* `cxxfunction` *points to NULL*

Description

The function object returned by `cxxfunction` cannot be saved across R sessions. This function can be used to see if we still have a valid function object. Also this function can be used for functions returned by `grab_cxxfun` of S4 class `cxxdso` since these functions are essentially created by `cxxfunction` or similarly.

Usage

```
is_null_cxxfun(cx)
```

Arguments

`cx`	A function of class `CFunc`

Details

R could not save the function objects that point to dynamically loaded functions, especially for those function created on the fly using package **inline** at least for one reason that those DSO's are deleted after quitting R. So it is always safe to tell if it is valid before call functions created by `cxxfunction`.

Value

Logical: `TRUE` null pointer; `FALSE`, not null, this function can still be called.

See Also

`cxxfunction`

Index