Package 'dir.expiry'

November 11, 2026

Version 1.20.0

Date 2024-10-17

Title Managing Expiration for Cache Directories

Description Implements an expiration system for access to versioned directories.
Directories that have not been accessed by a registered function within a certain time frame are deleted.
This aims to reduce disk usage by eliminating obsolete caches generated by old versions of packages.

License GPL-3

Imports utils, filelock

Suggests rmarkdown, knitr, testthat, BiocStyle

biocViews Software, Infrastructure

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

git_url https://git.bioconductor.org/packages/dir.expiry

git_branch RELEASE_3_23

git_last_commit b9df09a

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Aaron Lun [aut, cre]

Maintainer Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

Contents

clearDirectories 2
lockDirectory 3
touchDirectory 4

Index **7**

`clearDirectories` *Clear expired directories*

Description

Remove versioned directories that have passed on expiration limit.

Usage

```
clearDirectories(dir, reference = NULL, limit = NULL, force = FALSE)
```

Arguments

`dir`	String containing the path to a package cache containing any number of versioned directories.
`reference`	A package_version specifying a reference version to be protected from deletion.
`limit`	Integer scalar specifying the maximum number of days to have passed before a versioned directory expires.
`force`	Logical scalar indicating whether to forcibly re-examine `dir` for expired versioned directories.

Details

This function checks the last access date in the `*_dir.expiry` files in `dir` (generated by `touchDirectory`). If the last access date is too old, the corresponding subdirectory in `path` is treated as expired and is deleted. The age threshold depends on `limit`, which defaults to the value of the environment variable `BIOC_DIR_EXPIRY_LIMIT`. If this is not specified, it is set to 30 days.

If `reference` is specified, any directory of that name is protected from deletion. In addition, directories with version numbers greater than (or equal to) `reference` are not deleted, even if their last access date was older than the specified `limit`. This aims to favor the retention of newer versions, which is generally a sensible outcome when the aim is to stay up-to-date.

This function will acquire exclusive locks on the package cache directory and on each versioned directory before attempting to delete the latter. Applications can achieve thread safety by calling `lockDirectory` prior to any operations on the versioned directory. This ensures that `clearDirectories` will not delete a directory in use by another process, especially if the latter might update the last access time.

By default, this function will remember the values of `dir` that were passed in previous calls, and will avoid re-examining those same `dirs` for expired directories on the same day. This avoids unnecessary file system queries and locks when this function is repeatedly called. Advanced users can force a re-examination by setting `force=TRUE`.

Value

Expired directories are deleted and `NULL` is invisibly returned.

Author(s)

Aaron Lun

See Also

touchDirectory, which calls this function automatically when clear=TRUE.

Examples

```
# Creating the package cache.
cache.dir <- tempfile(pattern="expired_demo")

# Creating an older versioned directory.
version <- package_version("1.11.0")
version.dir <- file.path(cache.dir, version)

lck <- lockDirectory(version.dir)
dir.create(version.dir)
touchDirectory(version.dir, date=Sys.Date() - 100)
unlockDirectory(lck, clear=FALSE) # manually clear below.

list.files(cache.dir)

# Clearing them out.
clearDirectories(cache.dir)
list.files(cache.dir)
```

---

lockDirectory *Lock and unlock directories*

Description

Lock and unlock the package and version directories for thread-safe processing.

Usage

```
lockDirectory(path, ...)

unlockDirectory(lock.info, clear = TRUE, ...)
```

Arguments

path	String containing the path to a versioned directory. The dirname should be the package cache while the basename should be a version number.
...	For lockDirectory, further arguments to pass to lock. For unlockDirectory, further arguments to pass to clearDirectories.
lock.info	The list returned by lockDirectory.
clear	Logical scalar indicating whether to remove expired versions via clearDirectories.

Details

lockDirectory actually creates two locks:

- The first "V" lock is applied to the versioned directory (i.e., basename(path)) within the package cache (i.e., dirname(path)). This provides thread-safe read/write on its contents, protecting against other processes that want to write to the same versioned directory. If the caller is only reading from path, they can set exclusive=FALSE in ... to define a shared lock for concurrent reads across multiple processes.
- The second "P" lock is applied to the package cache and is always a shared lock, regardless of the contents of .... This provides thread-safe access to the lock file used by the V lock, protecting it from deletion when the relevant directory expires in clearDirectories.

If dirname(path) does not exist, it will be created by lockDirectory.

clearDirectories is called in unlockDirectory as the former needs to hold an exclusive lock on the package cache. Thus, the clearing can only be performed after the P lock created by lockDirectory is released.

Value

lockDirectory returns a list of locking information, including lock handles generated by the **filelock** package.

unlockDirectory unlocks the handles generated by lockDirectory. If clear=TRUE, versioned directories that have expired are removed by clearDirectories. It returns a NULL invisibly.

Author(s)

Aaron Lun

Examples

```
# Creating the relevant directories.
cache.dir <- tempfile(pattern="expired_demo")
version <- package_version("1.11.0")

handle <- lockDirectory(file.path(cache.dir, version))
handle
unlockDirectory(handle)

list.files(cache.dir)
```

---

touchDirectory *Touch a versioned directory*

Description

Touch a versioned directory to indicate that it has been successfully accessed in the recent past.

Usage

```
touchDirectory(path, date = Sys.Date(), force = FALSE)
```

Arguments

path	String containing the path to a versioned directory. The dirname should be the package cache while the basename should be a version number.
date	A Date object containing the current date. Only provided for testing.
force	Logical scalar indicating whether to forcibly update the access date for path.

Details

This function is used to update the last-accessed timestamp for a particular versioned directory. The timestamp is stored in a *_dir.expiry stub file inside path, and is checked by clearDirectories to determine whether the directory is old enough for deletion. We use an explicit timestamp instead of relying on POSIX access times as the latter may not be reliable indicators of genuine access (e.g., during filesystem scans for viruses or to create backups) or may be turned off altogether for performance reasons. The touchDirectory function should be used in the following manner to ensure thread safety:

1. The user should first call lockDirectory(path) before calling touchDirectory(path). This ensures that another process calling clearDirectories does not delete path while its access time is being updated.
2. The caller should then perform the desired operations on path. This may be a read-only operation if the V lock is a shared lock or a read/write operation for an exclusive lock.
3. Once the operation has completed successfully, the caller should call touchDirectory(path). This ensures that the most recent timestamp is recorded, especially for long-running operations. Multiple processes may safely call touchDirectory(path) when the V lock is shared.
4. Finally, the user should call unlockDirectory(path). This is typically wrapped in an on.exit to ensure that the locks are removed.

For a given path and version, this function only modifies the files on its first call. All subsequent calls with the same two arguments, in the same R session, and on the same day will have no effect. This avoids unnecessary filesystem writes and locks when this function is repeatedly called. Advanced users can force an update by setting force=TRUE.

Value

The <version>_dir.expiry stub file within path is updated/created with the current date. A NULL is invisibly returned.

Author(s)

Aaron Lun

See Also

lockDirectory, which should always be called before this function.

Examples

```
# Creating the package cache.
cache.dir <- tempfile(pattern="expired_demo")
dir.create(cache.dir)

# Creating the versioned subdirectory.
version <- package_version("1.11.0")
version.dir <- file.path(cache.dir, version)
lck <- lockDirectory(version.dir)
dir.create(version.dir)

# Setting the last access time.
touchDirectory(version.dir)
list.files(cache.dir)
readLines(file.path(cache.dir, "1.11.0_dir.expiry"))

# Making sure we unlock it afterwards.
unlockDirectory(lck)
```

Index

clearDirectories, 2, *3–5*

Date, *5*

lock, *3*
lockDirectory, *2*, *3*, 3, *5*

on.exit, *5*

package_version, *2*

touchDirectory, *2*, *3*, 4

unlockDirectory, *5*
unlockDirectory *(*lockDirectory*)*, 3