Package 'dir.expiry'

May 10, 2024

Version 1.12.0

Date 2022-09-04

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 7.1.1

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

git_branch RELEASE_3_19

git_last_commit 141ea0d

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-10

Author Aaron Lun [aut, cre]

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

Contents

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. 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

Mark directories as locked or unlocked for thread-safe processing, using a standard naming scheme for the lock files.

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 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. Concurrent read operations are also permitted by setting exclusive=FALSE in ... to define a shared lock..
- The second 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 in the first 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 shared 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 should be called *after* any successful access to the contents of a versioned directory, to indicate that said directory is still in use by expiry-aware processes. A stub file is updated with the last access time to allow `clearDirectories` to accurately check for staleness.

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 touching of the file system during routine use.

The caller should lock the target directory with `lockDirectory` before calling this function. This ensures that another process calling `clearDirectories` does not delete this directory while its access time is being updated. If the target directory is locked, any writes to the stub file itself are thread-safe, even for shared locks.

By default, this function will remember the values of `path` that were passed in previous calls, and will avoid re-updating those same `path`s with the same date when called on the same day. This avoids unnecessary file system 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