build passing Dependency Status devDependency Status

The UNIX command `rm -rf` for node.

Install with `npm install rimraf`, or just drop rimraf.js somewhere.

API

`rimraf(f, [opts], callback)`

The first parameter will be interpreted as a globbing pattern for files. If you want to disable globbing you can do so with `opts.disableGlob` (defaults to `false`). This might be handy, for instance, if you have filenames that contain globbing wildcard characters.

The callback will be called with an error if there is one. Certain errors are handled for you:

- Windows: `EBUSY` and `ENOTEMPTY` - rimraf will back off a maximum of `opts.maxBusyTries` times before giving up, adding 100ms of wait between each attempt. The default `maxBusyTries` is 3.
- `ENOENT` - If the file doesn't exist, rimraf will return successfully, since your desired outcome is already the case.
- `EMFILE` - Since `readdir` requires opening a file descriptor, it's possible to hit `EMFILE` if too many file descriptors are in use. In the sync case, there's nothing to be done for this. But in the async case, rimraf will gradually back off with timeouts up to `opts.emfileWait` ms, which defaults to 1000.

options

- unlink, chmod, stat, lstat, rmdir, readdir, unlinkSync, chmodSync, statSync, lstatSync, rmdirSync, readdirSync

  In order to use a custom file system library, you can override specific fs functions on the options object.

  If any of these functions are present on the options object, then the supplied function will be used instead of the default fs method.

  Sync methods are only relevant for `rimraf.sync()`, of course.

  For example:

  ```
  var myCustomFS = require('some-custom-fs')

  rimraf('some-thing', myCustomFS, callback)
  ```

- maxBusyTries

  If an `EBUSY`, `ENOTEMPTY`, or `EPERM` error code is encountered on Windows systems, then rimraf will retry with a linear backoff wait of 100ms longer on each try. The default maxBusyTries is 3.

  Only relevant for async usage.

- emfileWait

  If an `EMFILE` error is encountered, then rimraf will retry repeatedly with a linear backoff of 1ms longer on each try, until the timeout counter hits this max. The default limit is 1000.

If you repeatedly encounter `EMFILE` errors, then consider using graceful-fs in your program.

Only relevant for async usage.

- glob

  Set to `false` to disable glob pattern matching.

  Set to an object to pass options to the glob module. The default glob options are `{ nosort: true, silent: true }`.

  Glob version 6 is used in this module.

  Relevant for both sync and async usage.

- disableGlob

  Set to any non-falsey value to disable globbing entirely. (Equivalent to setting `glob: false`.)

rimraf.sync

It can remove stuff synchronously, too. But that's not so good. Use the async API. It's better.

CLI

If installed with `npm install rimraf -g` it can be used as a global command `rimraf <path> [<path> ...]` which is useful for cross platform support.

mkdirp

If you need to create a directory recursively, check out mkdirp.