



Lab 13.2 - Watching

The labs-2 folder contains an `index.js` file with the following:

```
'use strict'
const assert = require('assert')
const { join } = require('path')
const fs = require('fs')
const { promisify } = require('util')
const timeout = promisify(setTimeout)
const project = join(__dirname, 'project')
try { fs.rmdirSync(project, {recursive: true}) } catch (err) {
  console.error(err)
}
fs.mkdirSync(project)

let answer = ''

async function writer () {
  const { open, chmod, mkdir } = fs.promises
  const pre = join(project, Math.random().toString(36).slice(2))
  const handle = await open(pre, 'w')
  await handle.close()
  await timeout(500)
  exercise(project)
  const file = join(project, Math.random().toString(36).slice(2))
  const dir = join(project, Math.random().toString(36).slice(2))
  const add = await open(file, 'w')
  await add.close()
  await mkdir(dir)
```

```
    await chmod(pre, 0o644)
    await timeout(500)
    assert.strictEqual(
      answer,
      file,
      'answer should be the file (not folder) which was added'
    )
    console.log('passed!')
    process.exit()
  }

  writer().catch((err) => {
    console.error(err)
    process.exit(1)
  })

function exercise (project) {
  const files = new Set(fs.readdirSync(project))
  fs.watch(project, (evt, filename) => {
    try {
      const filepath = join(project, filename)
      const stat = fs.statSync(filepath)

      // TODO - only set the answer variable if the filepath
      // is both newly created AND does not point to a directory

      answer = filepath
    } catch (err) {

    }
  })
}
```

When executed (e.g. using `node index.js`) this code will create a folder named **project** (removing it first if it already exists and then recreating it), and then perform some file system manipulations within the **project** folder.

The **writer** function will create a file before calling the **exercise** function, to simulate a pre-existing file. The **exercise** function will then be called which sets up a file watcher with **fs.watch**. The **writer** function then proceeds to create a file, a directory and changes the

permissions of the previously existing file. These changes will trigger the listener function passed as the second argument to `fs.watch`.

The goal is to ensure that the `answer` variable is set to the newly created file. So when a directory is added, the `answer` variable should not be set to the directory path. When the preexisting files status is updated via a permissions change, the `answer` variable should not be set to that preexisting file.

If implemented correctly the process will output: **passed!**