Versioning

Due: October 26, 2015, 11:59:59 pm

v0.2

Changes since original:

- **v0.2:** specified that file archives do not need to include the “current” version.

1 Introduction

By default, your last project should be saving all data blocks ever used, as well as multiple versions of directory and file nodes. This project will leverage that data to provide access to previous versions of files and directories, i.e., become a *versioning file system.*

You should allow creation of *file archives* and *directory archives.* Both types of archives are created through `mkdir` calls and removed by `rmdir` calls. They are:

1. distinguished by having an ’@’ symbol in their names
2. read-only, (both files, and the new directories themselves)
3. are not persisted to disk.
4. are created in the same directory as their “base” file or directory

2 Tasks

2.1 Directory Archives

Directory archive names have two formats:

```
foo@2015-09-18 9:20
   and
foo@-1m
```

Both archives show a version of the “foo” directory. The first uses a time string that can be parsed via the `peteTime()` function in the included file `y.go`. The second uses a duration that can be parsed by `time.ParseDuration()` and should be added to the current wall clock time. Since you are using `ParseDuration()`, you should be able to use offsets in terms of seconds, or hours, as well.

Your file system should handle the following script:

```
mkdir foo
echo "nice" > foo/f1
sleep 30
cp foo/f1 foo/f2
sleep 30
cp foo/f1 foo/f3
sleep 30
ls -l foo
```

```
        mkdir "foo@-70s"

        set x = `date "+%s"`

        @ x2 = $x - 70
        set x2date = `date -d "1970-01-01 $x2 sec GMT" "+%Y-%m-%d %H:%M:%S"`
        mkdir "foo@$x2date"
        ls  "foo@$x2date"

        @ x1 = $x - 40
        set x1date = `date -d "1970-01-01 $x1 sec GMT" "+%Y-%m-%d %H:%M:%S"`
        mkdir "foo@$x1date"
        ls  "foo@$x1date"
```

Note:

- this script only works with `date` on linux systems, now macs
- only with `tcsh`, not `bash`

The first and second directory archives should have a single file, and the last should have two. The output of a "ls -l */f*" command should be something like:

```
hyperion:~/z> ls -l */f*
-rw-r--r-- 1 keleher keleher 5 Sep 18 12:16 foo@2015-09-18 12:17/f1
-rw-r--r-- 1 keleher keleher 5 Sep 18 12:16 foo@2015-09-18 12:18/f1
-rw-r--r-- 1 keleher keleher 5 Sep 18 12:17 foo@2015-09-18 12:18/f2
-rw-r--r-- 1 keleher keleher 5 Sep 18 12:16 foo@-70s/f1
-rw-r--r-- 1 keleher keleher 5 Sep 18 12:16 foo/f1
-rw-r--r-- 1 keleher keleher 5 Sep 18 12:17 foo/f2
-rw-r--r-- 1 keleher keleher 5 Sep 18 12:18 foo/f3
```

2.2 File Archives

File archives contain every saved version of a particular file, and are distinguished by a "@versions" suffix. Your code should support the following script:

```
        echo "nice" > food
        cp food bar
        sleep 10
        cat food >> bar
        sleep 10
        cat food >> bar
        sleep 10
        mkdir bar@versions
```

Assuming your data is being flushed every five seconds, or synchronously, your output should be something like the following:

```
hyperion:~/z> ls -l bar@versions
-rw-r--r-- 1 keleher keleher  5 Sep 18 12:34 bar.2015-09-18 12:34:05
-rw-r--r-- 1 keleher keleher 10 Sep 18 12:34 bar.2015-09-18 12:34:15
-rw-r--r-- 1 keleher keleher 15 Sep 18 12:34 bar.2015-09-18 12:34:25
```

Note that this only includes non-current versions, i.e. files that have been written to the object store by the flusher.

2.3 Something Else

Says it all. Overload the file system with some other semantically meaningful functionality.

3 Files

The following archive contains `time.go` and the two scripts above:

http://triffid.cs.umd.edu/dss/projects/p3.tgz

`time.go` has some new time manipulation examples.

4 Submit

You should include a `README` or `README.md` describing your any caveats, as well as the meaning and syntax of your new functionality.

Use the submit server as usual.